НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК *519.854.2*

«До захисту допущено»
Завідувач кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*18* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *122 Комп'ютерні науки та інформаційні технології*
(код та назва спеціальності)

спеціалізацією *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Задача маршрутизації безпілотних літальних апаратів*

Виконав: студент *VI* курсу групи ІС-63м
(шифр групи)

Рибальченко Олег Валерійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник *проф., д.т.н., с.н.с. Гуляницький Л. Ф.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

РЕФЕРАТ

Актуальність. Важливість використання безпілотних літальних апаратів (далі - БПЛА) у наш час важко переоцінити, особливо в умовах бойових дій. Можливість проведення розвідки та відеозйомка об’єктів супротивника без ризику для людського життя надає значну перевагу при плануванні операцій. У даній дисертації буде розглянуто ситуацію, за якої певна кількість БПЛА, що мають визначені характеристики, розміщується на різних базах та здійснюють вильоти для зйомки визначених територій. Кінцевою метою такої операції буде отримання знімків цих територій та мінімізація ризику втрати апаратів.

Важливим аспектом задачі є те, що керування літальними апаратами у реальному часі переважно є неможливим через навмисні перешкоди для радіозв’язку. Це означає, що операція має бути повністю спланована заздалегідь і БПЛА повинні мати повну програмну інструкцію для автоматичного здійснення польоту.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась у філії кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках науково-дослідної теми Інституту кібернетики ім. В. М. Глушкова НАН України: «Розробити математичний апарат, орієнтований на створення інтелектуальних інформаційних технологій розв’язування проблем комбінаторної оптимізації та інформаційної безпеки» (шифр теми: ВФ.180.11).

Метою роботи є зниження витрат енергії при виконанні операції безпілотними літальними апаратами, запобігання їх потраплянню у радіус дії загроз супротивника, підвищення відсотку розвіданої місцевості

внаслідок проведення операції, зменшення тривалості операції. **Об'єктом дослідження** є операція проведення розвідки за допомогою БПЛА, **предметом дослідження** - попереднє планування такої операції.

Завдання дослідження:

- провести аналіз потреб у плануванні польотів БПЛА;
- розробити математичний апарат для розв'язку задачі їх маршрутизації з урахуванням характерної для предметної області обмежень;
- дослідити результати планування з використанням математичного апарату та зробити висновок про доцільність його подальшого використання;
- розробити програмний комплекс для автоматизації створення плану польотів.

Наукова новизна одержаних результатів полягає у розробці планувальника операції обльоту вказаних цілей із застосуванням одного або більше БПЛА та можливістю використання багатьох депо, а також у розробці та порівняльному аналізі нового алгоритму, що базується на методі оптимізації мурашиними колоніями.

КАРТОРАФІЯ, МАРШРУТИЗАЦІЯ, БЕЗПІЛОТНИЙ ЛІТАЛЬНИЙ АПАРАТ, УНИКНЕННЯ ЗАГРОЗ, ПЛАНУВАННЯ ОПЕРАЦІЇ, VRP

ЗМІСТ

ВСТУП

Важливість використання безпілотних літальних апаратів (далі - БПЛ) у наш час важко переоцінити, особливо в умовах бойових дій. Можливість проведення розвідки та відеозйомка об’єктів супротивника без ризику для людського життя надає значну перевагу при плануванні операцій. У даній дисертації буде розглянуто ситуацію, за якої певна кількість БПЛ, що мають визначені характеристики, розміщується на різних базах та здійснюють вильоти для зйомки визначених територій. Кінцевою метою такої операції буде отримання знімків цих територій та мінімізація ризику втрати апаратів. Планування операції покладається на програмний комплекс, постановка задачі якого буде наведена далі. Актуальність обраної теми в Україні, зокрема, демонструють деякі тендерні закупівлі військових частин та державних підприємств [13][14].

Важливим аспектом задачі є те, що керування літальними апаратами у реальному часі переважно є неможливим через навмисні перешкоди для радіозв’язку. Це означає, що операція має бути повністю спланована заздалегідь і БПЛ повинні мати повну програмну інструкцію для автоматичного здійснення польоту. Наявність великої кількості можливих точок запуску та баз для здійснення посадки ускладнює ручне планування операції, знижує його оперативність та точність. Саме тому виникає необхідність у створенні автоматизованої системи попереднього планування операції. Зокрема, це дозволить знизити витрати на пальне та запобігти потраплянню БПЛ у радіус дії загроз з боку супротивника.

Також слід зазначити, що такий планувальник без внесення суттєвих змін може бути задіяний для багатьох інших задач. Наприклад, задачі збирання інформації з датчиків для моніторингу стану ґрунту на посівних площах чи задачі планування роботи групи комбайнів для

збирання врожаю. Велика кількість різноманітних задач може бути зведена до аналогічної постановки, а отже, розв’язана таким планувальником. Також було здійснено огляд літератури, проаналізовано існуючі наукові розробки у предметній області та методи розв’язання схожих задач. В сукупності з наведеним вище це дозволяє зробити висновок про актуальність дослідження та створення автоматизованої системи.

Багато часткових результатів українських та міжнародних досліджень може бути використано при розробці математичної бази планувальника операцій, зокрема, у дослідженнях маршрутизації цивільної авіації [1]. Такі задачі активно розглядаються в іноземних наукових роботах, світові тенденції у їх вирішенні полягають у зведенні до класичної задачі маршрутизації транспортних засобів [7][8]. Огляд таких досліджень та їх результатів дозволив урахувати переваги та недоліки існуючих методів при розробці системи.

Метою роботи є зниження витрат енергії при виконанні операції безпілотними літальними апаратами, запобігання їх потраплянню у радіус дії загроз супротивника, підвищення відсотку розвіданої місцевості внаслідок проведення операції, зменшення тривалості операції. Об’єктом дослідження є операція проведення розвідки за допомогою БПЛ, предметом дослідження - попереднє планування такої операції.

Завдання дослідження:

- провести аналіз потреб у плануванні польотів БПЛ;
- розробити математичний апарат для розв’язку задачі їх маршрутизації з урахуванням характерної для предметної області обмежень;

- дослідити результати планування з використанням математичного апарату та зробити висновок про доцільність його подальшого використання;
- розробити програмний комплекс для автоматизації створення плану польотів.

1 ОГЛЯД ІСНУЮЧИХ ДОСЛІДЖЕНЬ

Найбільш повним з розглянутих досліджень можна вважати роботу Kamil A. Alotaibi - “Unmanned Aerial Vehicle Routing In The Presence Of Threats” [1]. У ній розглядаються основні принципи дії попереднього планувальника операції (pre-mission planner) в умовах наявності точок, що необхідно відвідати за умови мінімізації витрат та ризику потрапляння у радіус дії загроз. Цікавим також є те, що автор розглянув навіть вплив вітру на літальні апарати, хоч і незмінного в часі. Особлива цінність роботи полягає у наведенні різних способів вибору проміжних точок для зведення задачі до класичної задачі маршрутизації (Vehicle Routing Problem, VRP). Також розглядаються детерміновані та недетерміновані методи знаходження плану.

У статті розглядається така постановка задачі, за якої планувальник операції отримує інформацію про місцевість, включаючи місця розташування цілей, загроз, депо та БПЛА. Потім планувальник формує набір проміжних точок, які БПЛА може відвідувати при перельоті між цілями та депо з метою зниження ризику при потраплянні у радіус дії загроз. Планувальник має заданий рівень допустимого ризику, залежно від якого формує один або декілька оптимізованих маршрутів.

На рисунку 1.1 наведено приклад графічного зображення однієї з розглянутих у статті задач та оптимізованого шляху відвідування цілей з урахуванням зниження ризику потрапляння у радіус дії загроз. На поданій схемі використовуються наступні позначення:

- синій ромб – побудована проміжна точка для оминання загроз;
- червоний трикутник – загроза;
- зелений круг – ціль, яку необхідно відвідати;
- чорний круг – база вильоту та прильоту БПЛА;

– лінії без штриховки зі стрілками визначають напрямки перельотів між цілями та проміжними точками;

– штрих-пунктирна лінія визначає прямий переліт, що було б здійснено без урахування загрози.

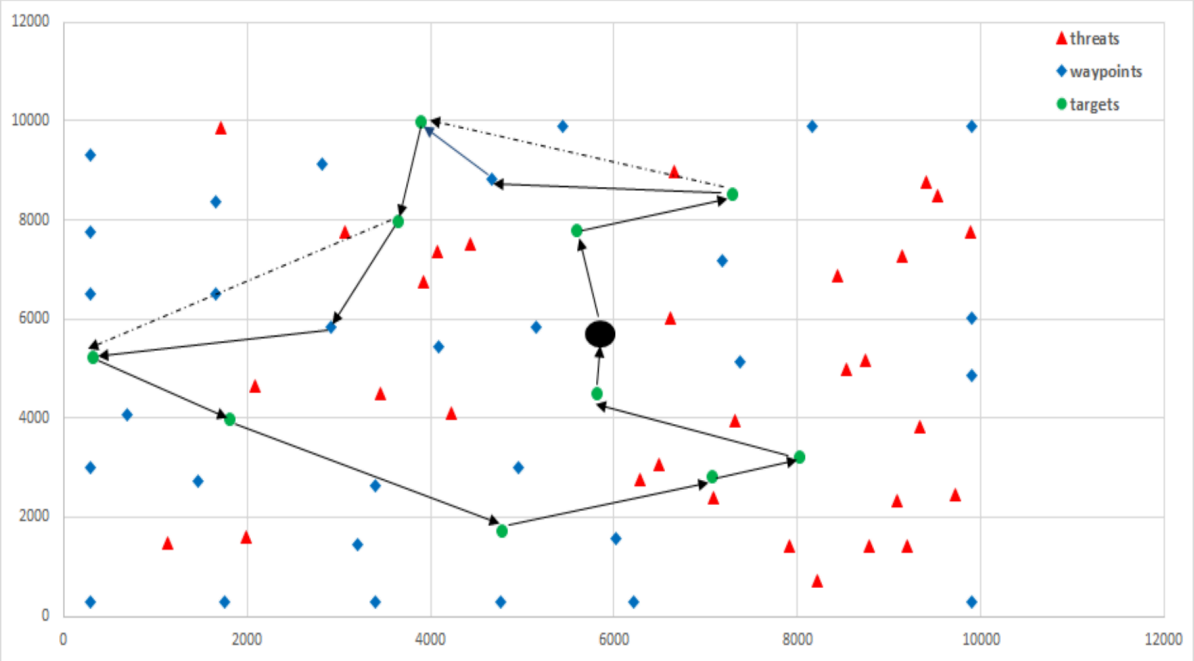


Рисунок 1.1 – Оптимізація маршруту з урахуванням загроз

Одним з наведених у роботі методів побудови проміжних точок доцільно скористатися при подальшому розвитку програмного комплексу, описаного в рамках даної дисертації. Для кожної пари з набору цілей та баз виконуємо наступні дії:

– вписуємо в квадрат таким чином, щоб вершини були на кінцях його діагоналі;

– перевіряємо чи потрапляє хоч одна точка найкоротшого шляху між ними у радіус дії загрози;

– якщо потрапляє, то для кожної такої загрози:

– обираємо по одній точці на кожній з двох прямих між загрозою та кутом квадрату, що не відповідає обраним вершинам. Критерій вибору -

найближча до загрози точка, відрізки між якою та обраними вершинами не потрапляють в радіус дії даної загрози;

– додаємо обрані точки до початкового набору з метою їх подальшої аналогічної обробки.

Після виконання таких операцій отримаємо набір додаткових проміжних точок, завдяки яким можна буде оминути загрозу при перельоті між будь-якою парою вершин.

Відмінність розглянутої у статті [1] задачі від тої, що розглядається у рамках даної дисертації полягає у:

– використовуваному алгоритмі;

– відсутності можливості відправлення БПЛА з багатьох депо;

– неможливості заміни блоків живлення БПЛА у депо.

Найбільше це джерело задіяне при розробці математичної постановки, оскільки схожа задача формалізована в ході поданого дослідження.

Схожа задача у значно спрощеному вигляді описана у статті [3]. В ній також розглядається постійний вплив вітру, проте немає загроз та проміжних точок. В цій статті депо може бути одне (маршрут замкнений) або два (маршрут розімкнений). На рисунку 1.2 показано приклад зі статті [2а] для побудови маршруту, у якому початкова та кінцева точка відрізняються.

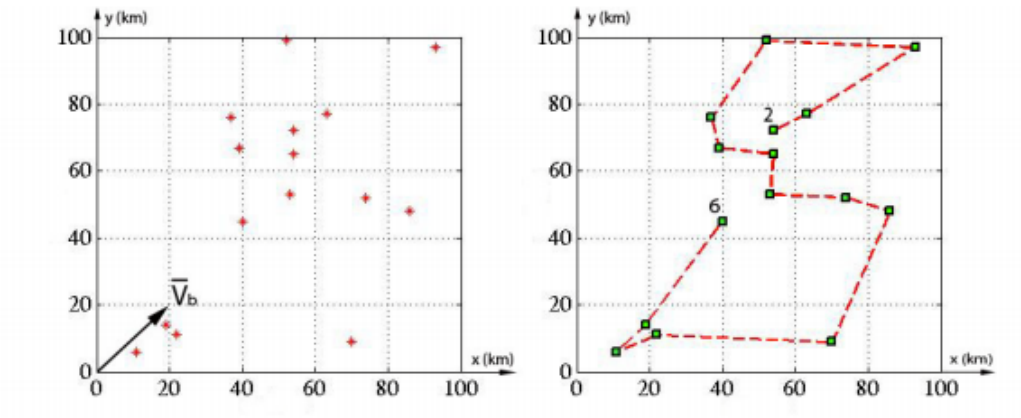


Рисунок 1.2 – маршрут, початкова та кінцева точки якого не співпадають.

На рисунку 1.3 зображено приклад зі статті, для якого початок і кінець маршруту збігається.

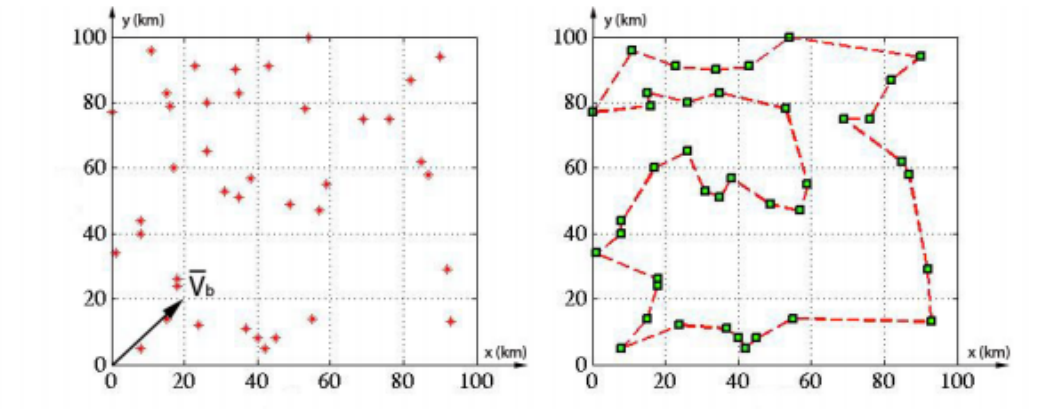


Рисунок 1.3 – маршрут з одним депо

Задача, розглянута у статті [3], відрізняється від розглянутої у даній дисертації наступними аспектами:

- використано метод гілок та меж, що не дозволяє застосовувати програму для наборів даних з великою кількістю точок;
- розглянута маршрутизація лише одного БПЛА, що є частковим випадком задачі комівояжера;

– відсутня повноцінна підтримка багатьох депо з можливістю вильоту та прильоту.

Задачу маршрутизації БПЛА буде зведено до добре відомої задачі маршрутизації транспорту VRP, розглянутої далі. Розподілення товарів є однією з ключових функцій у логістиці та включає їх доставку від виробника до споживача через транспортну мережу. Такі переміщення вимагають великих витрат, особливо для галузі доставки. VRP - загальна назва класу задач комбінаторної оптимізації, в яких споживачі обслуговуються певною кількістю машин. Машини базуються у депо, виїжджають для обслуговування споживачів (доставки), після чого повертаються. Існує велика кількість різноманітних варіантів постановок задачі - з обмеженнями на тривалість, час доставки, вантажопідйомність машин, максимальну довжину шляху тощо.

Класична задача маршрутизації транспорту задається орієнтованим графом *G(E,V)*, де *V={0, 1, … , n}* задає набір транспортних вузлів (вершин), E - множину ребер. Депо (база) записується як вузол *j=0*, а споживачі *j = 1, 2, .. , n*, кожен з яких характеризується потребою $d_j > 0$. Кожне ребро відповідає шляху з вершини i до вершини j. Вага кожного ребра $C_{ij} > 0$ відповідає вартості (часу, відстані тощо) переїзду з *i* до *j*. Якщо $C_{ij} = C_{ji}$, задача є симетричною, інакше - асиметричною. З точки зору складності, VRP відноситься до класу NP-складних задач, оскільки узагальнює задачу комівояжера (далі - TSP) та задачу про пакування контейнерів (далі - BPP), що є добре відомими NP-складними задачам [7]. Математичне формулювання класичної VRP можна знайти у роботі [8]. Багато робіт присвячено як класичній версії VRP, так і її модифікаціям. Існує достатньо велика кількість задач VRP [4-7], які відрізняються одна від одної врахуванням тих або інших характеристик вузлів, транспортних

засобів, а також введенням додаткових обмежень. Наявність різних обмежень визначається типом задач, а часто також тим, що реальні задачі є дуже складними і в таких випадках у першу чергу розглядаються спрощені задачі. Найпростішим варіантом VRP є задача з одним складом та однотипними транспортними засобами (Capacitated VRP). До обмежень задач VRP відносяться:

- наявність часових вікон вузлів (VRP with Time Windows);
- використання декількох типів транспортних засобів (VRP with Heterogeneous Fleet);
- врахування різних типів вузлів, наприклад, коли в одному вузлі необхідно забрати певні товари і доставити їх згодом в інший вузол (VRP with Pickup and Delivery);
- необхідно забрати товари та доставити їх на склад (VRP with Backhauls);
- кожний вузол-клієнт може обслуговуватись тільки транспортними засобами певного типу (Site Dependent VRP).

Особливості деяких з них є актуальними у контексті UAVRP, розглянемо більш детально деякі з них:

- з обмеженням на вантажопідйомність (Capacitated VRP, далі – CVRP) – обсяг вантажу на кожному маршруті не повинен перевищувати граничного значення [9];
- з часовими вікнами (VRP with Time Windows, далі – VRPTW) – для відвідування вершин існує певний допустимий проміжок часу [10];
- з багатьма депо (Multiple Depot VRP, далі – MDVRP) – відправлення може здійснюватися з багатьох депо [11];
- з різноманітним транспортом (Split Delivery VRP, далі – SDVRP) – характеристики транспортних засобів можуть відрізнятися [12].

Розрізняють два класи проблем: P та NP. Ці класи можна визначити точніше за допомогою будь-якого формального означення алгоритмів, такого, наприклад, як машина Тьюрінга. Не заглиблюючись у більш формальний опис, надалі нам буде достатньо означити їх так.

Клас P складають задачі, для розв'язування яких відомі алгоритми з поліноміальною складністю (polynomial-time algorithms). До поліноміальних відносять алгоритми, складність (трудомісткість) яких обмежена поліномом, степінь якого залежить від розміру входу ЗКО. Отже, це клас відносно простих задач, для яких існують ефективні алгоритми.е

До класу NP (nondeterministic polynomial), який здається ширшим, належать задачі, що можуть бути розв'язані недетермінованим поліноміальним алгоритмом. Такі алгоритми мають дві фази: на першій знаходять припустимий варіант розв'язку, а на другій цей варіант перевіряється детермінованим поліноміальним алгоритмом верифікації.

Задача комбінаторної оптимізації n є NP-повною (NP-complete), якщо виконуються дві умови:

- $n \in NP$;
- усі задачі з NP можуть бути зведені до n за допомогою поліноміального алгоритму.

На практиці для верифікації зазвичай лише показують, що деяку відому NP-повну задачу можна поліноміально перетворити на досліджувану задачу. Інколи можна показати, що всі задачі з NP поліноміально зводяться до деякої задачі n, але не вдається довести, що $n \in NP$. Такі задачі відносять до NP-складних.

Для потреб практики важливо, щоб трудомісткість алгоритмів була обмежена поліномом від вхідних даних задачі. Нині предметом дискусій є питання, чи $P = NP$. У разі негативної відповіді теоретично не існує поліноміального алгоритму, тому при розв'язуванні таких задач одразу слід зосередитися на розробці потужних наближених алгоритмів, які, хоча й не гарантують отримання оптимального розв'язку, здатні знаходити оптимальний чи близький до нього розв'язок за прийнятний час [2].

За своєю складністю задача маршрутизації безпілотних літальних апаратів відноситься до класу NP-складних задач, тобто складність обчислення експоненційно залежить від обсягу початкових даних [7]. Для розв’язання цієї задачі використовують точні, евристичні та метаевристичні методи [8]. Точні методи розв’язання VRP зазвичай орієнтуються на її загальне формулювання, в якому пропонується симетрична чи несиметрична матриця відстаней між пунктами. Вони забезпечують пошук оптимального рішення, проте, через надмірне зростання тривалості обчислень при збільшенні кількості пунктів їх, зазвичай, неможливо застосовувати для задач з більш ніж 25–30 вершинами [8]. Евристичні або метаевристичні методи передбачають відносно обмежений пошук рішень, і, зазвичай, знаходять близьке до оптимального рішення значно швидше, порівняно з точними методами, проте їх ефективність знижується при наближенні до кінця обчислень [9]. Ефективністю алгоритму тут вважається здатність знаходити якомога коротші маршрути. Багато метаевристичних методів базується на спостереженнях за явищами живої і неживої природи. До них належать генетичний алгоритм, кліткові автомати, алгоритм імітації відпалу, алгоритм на основі мурашиних колоній, пошук з вилученнями та інші [12-14]. Оскільки VRP відноситься до класу NP-складних задач, дослідники часто змушені використовувати евристичні методи (Chiang і Russell[20];

Braysy та ін. [21]; Nagy and Salhi [22] і Choi and Tcha[23]). Проте точні алгоритми для VRP також застосовуються.

У цьому предметному напрямку було зроблено багато внесків, включаючи різноманітні доповнення до описаного вище основного завдання. Так, Лапортою (Laporte) в 1992 р. [8] був зроблений огляд, а в 1995 складено разом з Османом (Osman) [24] обширну бібліографію. Taillard [25] і Rochat і Taillard [26] застосували табу-пошук (Tabu Search, TS) для багатьох варіантів VRP, було отримано найкращі на той час результати для деяких контрольних задач. Різні автори повідомляли про аналогічні результати, отримані за допомогою алгоритмів TS чи імітації відпалу (Simulated Annealing SA). Разом з тим, такі евристики потребують значного часу на обчислення і налаштувань багатьох параметрів [26].

Класифікація модифікацій класичної VRP, методів їх розв’язання та огляд наукових робіт, присвячених кожній з них, наведено у роботі «A literature review on the vehicle routing problem with multiple depots» [15]. Описано поширені додаткові умови та обмеження, зроблено огляд наукових робіт та методів розв’язання за останні 37 років. Також важливим результатом цього дослідження є наявність формальних постановок розглянутих модифікацій задачі та їх математичних моделей. Посилання на відповідну літературу для кожної розглянутої модифікації полегшило аналіз існуючих наукових розробок і дозволило оглянути значно більшу кількість джерел. Також більш детальний розгляд підходів до розв’язання поданого класу задач здійснено у роботі “The Family of Vehicle Routing Problems”. Це дозволило краще ознайомитися з існуючими методами розв’язання задач маршрутизації транспортних засобів та врахувати їх основні переваги і недоліки при розробці алгоритму.

Оскільки на задачах великої розмірності застосування точних алгоритмів є практично недоцільним через експоненційно зростаючу кількість обчислень, у більшості випадків застосовуються наближені алгоритми комбінаторної оптимізації.

Необхідність розробки ефективних наближених АКО, які застосовуються у переважній більшості випадків на практиці, визначається низкою обставин [2]:

- практично всі важливі задачі належать до NP-складних, тож точне їх розв'язання дуже проблематичне навіть із використанням сучасних і перспективних комп'ютерів;
- їх цільові функції мають зазвичай велику кількість локальних екстремумів;
- у багатьох прикладних проблемах дані задаються з певними похибками, що робить недоцільними ті істотні обчислювальні затрати, які необхідні для знаходження їх точного розв'язку;
- покладені в основу розробки наближених обчислювальних схем ідеї (метаевристики) дозволяють створювати алгоритми, які можуть розв'язувати не одну, а цілий клас близьких за постановкою оптимізаційних задач;
- важливий клас оптимізаційних задач породжується проблемами з директивним терміном, тобто їх розв'язок має бути знайдений до зазначеного апріорі строку;
- у деяких задачах значення цільової функції можуть бути доступними лише в процесі розв'язання задачі або змінюватися з часом – цей клас утворюють динамічні, або on-line задачі.

Найуживаніші на практиці наближені алгоритми можна поділити на сім класів (рисунок 4). Тут МГіМ – це метод гілок і меж, ПАВ –

послідовний аналіз варіантів; ці точні алгоритми використовуються в даному контексті для породження наближених обчислювальних схем.

Зрозуміло, що сукупність усіх розроблених донині АКО значно перевищує перелік наведених на цьому рисунку. Проте, як свідчить аналіз наукових публікацій, саме зазначені алгоритми та їх модифікації є основним інструментом для розв'язання практичних ЗКО.

Левову частку алгоритмів із класів 2 -7 становлять ітераційні методи, багато з яких базуються на використанні процедур локального пошуку. Крім того, для розв'язання задач із підвищеною точністю значного поширення набули метаевристики.

Метаевристики класифікують відповідно до парадигми, яка використовується при їх побудові. У статті [4] підхід до класифікації та наведені приклади поширених метаевристик, серед яких такі.

Детермінований локальний пошук - це метаевристики ітераційного типу, в основі яких лежить частковий перебір варіантів на кожній ітерації в околі поточного розв'язку.

Виділяють такі поширені метаевристики цього типу:

- пошук зі змінюваними (пульсуючими) околами;
- керований локальний пошук;
- табу-пошук.

Метаевристики на основі стохастичного локального пошуку використовують у якості підлеглої процедуру локального пошуку, у яку вбудовано ймовірнісні механізми. Це, зокрема, дозволяє будувати нерелаксаційні обчислювальні схеми з метою уникнення передчасної збіжності. Найбільш відомі такі алгоритми:

- метод звужувальних околів;
- алгоритми імітаційного відпалу;
- О-алгоритми;
- повторюваний локальний пошук;
- квантовий відпал;
- метод кросс-ентропії;
- GRASP.

Еволюційні метаевристики - це алгоритми пошуку, що використовуються для розв'язання задач оптимізації і моделювання шляхом послідовного відбору, комбінування і варіації шуканих параметрів розв'язків із використанням механізмів, що нагадують біологічну еволюцію. Найбільш відомими є:

- генетичні алгоритми;
- міметичні алгоритми;
- імунні алгоритми.

Варто відзначити, що за певної інтерпретації складових компонент до цього класу можна віднести і ряд метаевристик, які описуються авторами у інших термінах.

Метаевристики на основі ройового інтелекту – децентралізовані системи простих агентів, які локально взаємодіють як з середовищем, так і між собою. Часто їх об'єднують під назвою "навіяні природою чи засновані на принципах біології" [5]. Ці Метаевристики бурхливо розвиваються в останні роки, тому важко скласти повний перелік. Відмітимо ті, які вже пройшли певний період досліджень і апробації:

- мурашині алгоритми;
- оптимізація роєм часток;

- бджолині алгоритми;
- алгоритми світлячків;
- алгоритми моделювання поведінки бактерій.

Останнім часом запропоновані і досліджуються також алгоритми комбінаторної оптимізації, засновані на моделюванні чи імітації поведінки косяків риб, жаб, летючих мишей.

Методи сканування простору - популяційні метаевристики, на кожній ітерації яких формуються напрями нелокального пошуку в просторі розв’язків на основі наявних декількох варіантів наближень.

Метою є диверсифікація процесу пошуку у просторі розв'язків. У комбінаторній оптимізації знайшли застосування такі методи цього типу:

- розсіяний пошук;
- перекомпоновка маршрутів;
- Н-метод.

Як випливає, метаевристики використовують різні стратегії як при виборі наступного поточного розв'язку, так і для прийняття такого розв'язку. Підсумовуючи сказане, перелічимо деякі із найважливіших особливостей цих стратегій:

- траєкторні методи. При пошуку наступного наближення пошук ведеться в околі поточного розв'язку. Це правило застосовується, наприклад, у порогових алгоритмах та методах табу;
- методи нелокального збурення. Здійснюють пошук у всьому просторі розв'язків. Прикладами є переходи у повторюваному локальному пошуку, генетичних алгоритмах, Н-алгоритмах і колоніях мурах;

- популяційні (мультиагентні) методи. Результати пошуку агентів (простих процедур чи алгоритмів) заносяться в колективний досвід;
- керований пошук чи пошук з використанням пам'яті. Включають ряд додаткових правил и вказівок щодо напряму пошуку. У генетичних і міметичних алгоритмах популяція містить пам'ять про останній крок пошуку; у мурашиних алгоритмах адаптивною пам'яттю всієї колонії є матриця маршрутів. В пошуку табу список заборон є короткостроковою пам'яттю;
- некерований пошук чи методи без запам'ятовування Провадять повністю евристичний пошук.

Окремо розглянемо алгоритми оптимізації мурашиною колонією (ОМК; ant colony optimization – ACO), що широко застосовуються для розв’язування VRP.

Мурашині алгоритми – це багатоагентні системи, де поведінка кожного агента, який називається штучною мурахою, або просто мурахою, заснована на поведінці справжніх мурашок. Вони успішно застосовуються для розв’язування багатьох типів задач комбінаторної оптимізації, починаючи з класичної задачі комівояжера.

В алгоритмах ОМК формується спеціальна модель задачі, що розв'язується, тому вони належать до класу моделе-орієнтованих методів.

Така модель задачі подається у вигляді повного зваженого графа $G(V,E)$, де $v_i \in V, i = 1,...,n,$ – вершини, що відповідають компонентам розв'язку, а $e_{ij} \in E, e_{ij} = (v_i, v_j), v_i, v_j \in V,$ – ребра, які відповідають можливим з'єднанням (переходам) між відповідними вершинами.

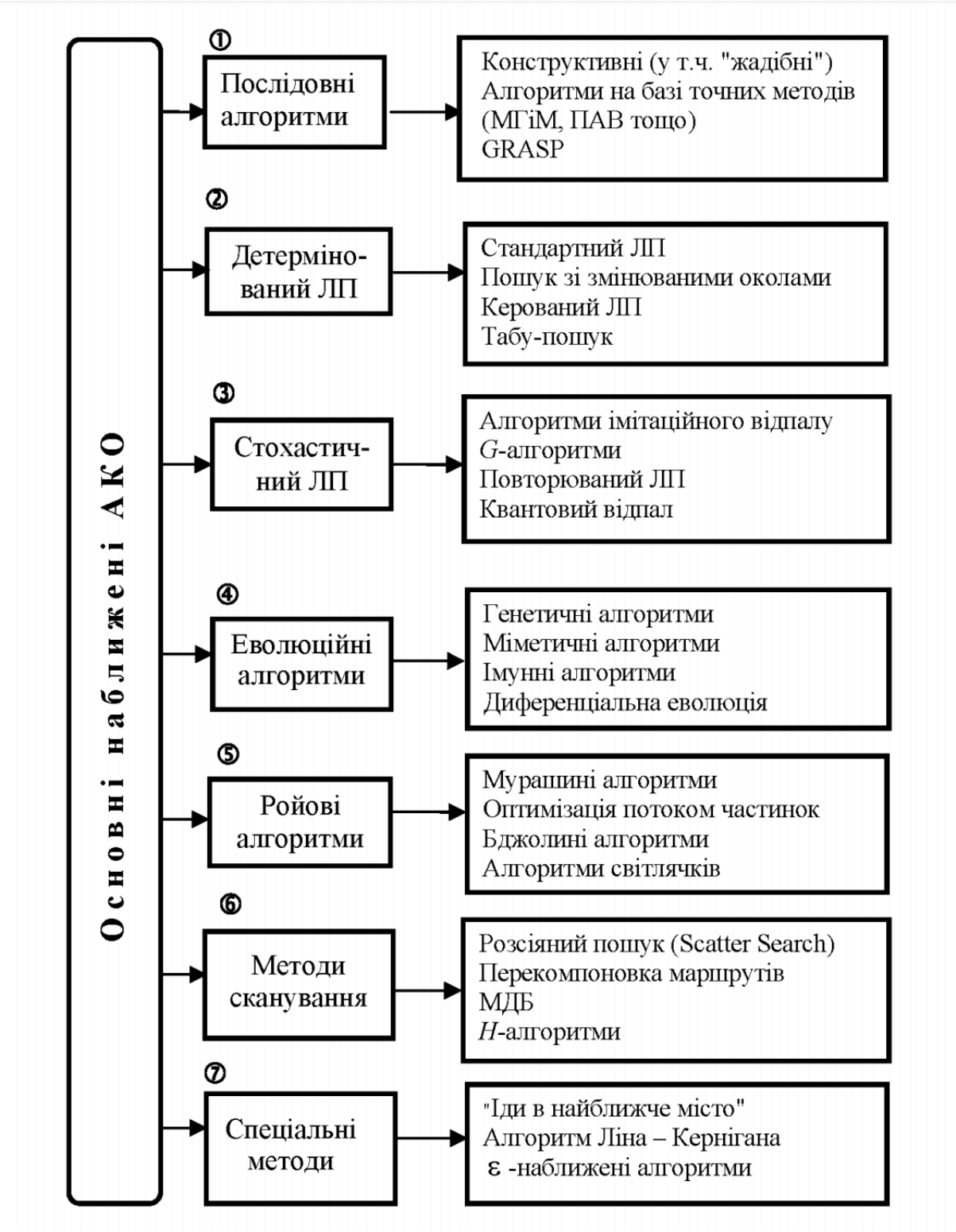


Рисунок 4 –класифікація основних наближених методів комбінаторної оптимізації [2]

Для кожного ребра може бути визначена функція вартості з'єднання, можливо, залежна від деякого параметра часу *t*.

Умови задачі, що розв'язується, можуть визначати набір обмежень $\Pi = \Pi(V, E, t)$ для елементів V та E, які визначають припустимість зав'язків між компонентами та з'єднаннями, а в підсумку – і побудованого з них розв'язку.

Розв'язки задачі оптимізації можуть бути подані як припустимі шляхи на графі G. Алгоритми ОМК можуть використовуватися для знаходження припустимих шляхів мінімальної вартості, що задовольняють обмеженням задачі. Вартість – значення цільової функції – $f(x)$, що відповідає розв'язку x, є функцією всіх вартостей з'єднань, які належать цьому розв'язку.

Мурахою в таких алгоритмах фактично є параметричний рандомізований жадібний алгоритм, який покроково будує з множини компонент (вершин чи ребер графа задачі) припустимий розв'язок задачі. У своїй роботі він використовує евристичну інформацію та феромонний слід, які характеризують ребра (рідше – вершини) графа задачі.

Евристична інформація (зазвичай позначається η_{ij}) – це числове значення, що не залежить від знайдених на попередніх кроках розв'язків і відображає ступінь бажаності включення в побудований фрагмент розв'язку того чи іншого нового ребра графа моделі $e_{ij} \in E$. Евристичні значення η_{ij} базуються на апріорній інформації, що відображає умови конкретної задачі та надається джерелом, відмінним від мурах; вони можуть залежати від часу (ітерації) t.

Рівень феромону (феромонний слід) – τ_{ij}, що відповідає ребру $e_{ij} \in E$, – це додатне число, яке показує, наскільки часто мурахами використовувалося це ребро на попередніх кроках чи при формуванні повного розв'язку. Феромонні сліди є для мурах довготривалою пам'яттю щодо всього процесу пошуку. Залежно від вибраного способу подання

задачі, феромонні сліди можуть відповідати всім дугам задачі або тільки деяким з них.

У мурашиних алгоритмах популяція агентів (або мурашок) спільно розв’язує сформульовану задачу оптимізації, використовуючи вищезазначене подання на графі моделі задачі [4].

В усіх відомих модифікаціях ОМК на кожному кроці для вершини, що є останньою в поточному фрагменті розв'язку (маршруту в графі задачі), формується множина припустимих сусідніх вершин і обчислюється ймовірність переходу до кожної з цих вершин. На основі цих ймовірностей і вибирається чергова вершина для продовження наявного фрагменту маршруту.

Далі пропонується підхід для формування маршруту, при якому мурахи на кожному кроці використовують інформацію не тільки з одного ребра, що може бути включене у фрагмент розв'язку, а також і інформацію з більшої кількості можливих ребер. Отже, на кожній ітерації мураха може додати до шляху одразу декілька вершин або, іншими словами, може зробити декілька «кроків».

Так для двокрокової версії алгоритму ОМК для кожної мурахи k за наявності уже побудованого фрагменту розв'язку $y=(\ldots,i)$ формується множина ще невідвіданих ребер (i,s), (s,j), $s \in N_i^k$, $j \in N_s^k$, де N_s^k – множина припустимих для мурахи k вершин графа задачі за умови, що до існуючого фрагмента розв'язку додана вершина s, $y^+=(\ldots,i,s)$, та обчислюється ймовірність переходу від вершини i до вершини j через вершину s з урахуванням евристичної інформації та поточних значень феромону. [4] Для двокрокового алгоритму ОМК перехід k-ї мурахи з вершини i в j через вершину s на поточній ітерації t пропонується здійснювати з ймовірністю, що може розраховуватися за наступною формулою:

$$p_{ij}^{\ k} = \frac{[\tau_{is}(t) \cdot \tau_{sj}(t)]^{\alpha} [\eta_{is}(t) + \eta_{sj}(t)]^{\beta}}{\sum_{r \in N_{ilj}^{\ k}} [\tau_{ir}(t) \cdot \tau_{rj}(t)]^{\alpha} [\eta_{ir}(t) + \eta_{rj}(t)]^{\beta}},$$

де $N_{ilj}^{\ k} = \{l : l \in N_i^{\ k},\ j \in N_l^{\ k}\}$.

Ці дослідження застосовні як для задачі комівояжера, так і для VRP, тож дозволяють застосувати новий підхід для задачі маршрутизації БПЛА. Деякі з цих принципів було покладено у основу нового алгоритму, що описується у даній дисертації.

З огляду літератури випливає, що дещо подібні задачі вже вирішувалися, проте велику кількість специфічних для обраної предметної області факторів розглянуто не було. Зокрема, неможливість постійного контролю літальних апаратів під час проведення операції, необхідність оминання загроз та можливість ручного коригування автоматично розробленого плану. Також у відкритому доступі не згадується програмний комплекс, що вирішує проблему планування наведених вище операцій. Отже, з урахуванням результатів аналізу існуючих робіт, можна стверджувати, що тема є актуальною.

2 МОДЕЛІ ТА МЕТОДИ ПОБУДОВИ МАРШРУТІВ

Маршрутом для кожного БПЛА називається послідовність точок, що відповідає наступним умовам:

- починається з депо, що придатне для здійснення вильотів;
- закінчується депо, що придатне для здійснення посадки;
- містить точки, що є цілями або депо;
- перельоти через цілі між початковим, кінцевим та проміжними депо зі здатністю заміни блоків живлення, за довжиною не перевищують максимальну дальність польоту БПЛА.

Внаслідок планування операції кожен БПЛА отримує призначення у вигляді допустимого маршруту, або відмічається як незадіяний.

Математична модель задачі

Планувальник операції отримує наступну інформацію:

n – кількість пускових точок (баз БПЛА), з яких можуть здійснюватися вильоти та які можуть приймати апарати;

m – кількість БПЛА;

m_{j0} – максимальна дальність польоту j-го БПЛА, км;

m_{j1} – швидкість польоту j-го БПЛА, км/год;

k – кількість цілей, що необхідно відвідати;

C_{ij} – відстань між точками i та j.

Окремо зазначимо, що застосування формули обчислення відстані між точками на двовимірній площині призведе до похибки через викривлення траекторії при переміщенні над поверхнею. Для розрахунку відстані по поверхні Землі використаємо формулу гаверсинуса – важливе рівняння у навігації, яке дозволяє обчислити відстань між точками на сфері, за їх довготою та широтою.

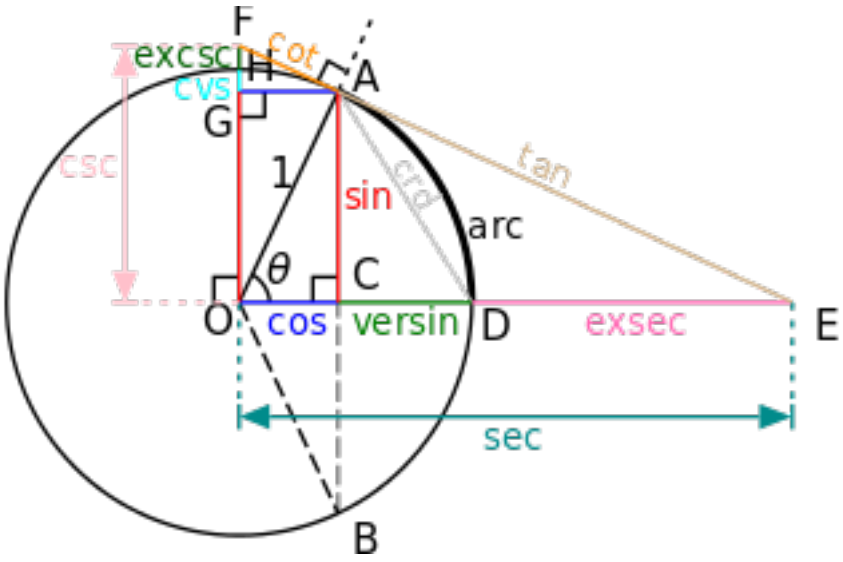


Рисунок 2.1 – одиничне коло з зображенням менш поширених тригонометричних співвідношень.

Є окремим випадком більш загальної формули сферичної тригонометрії, закону гаверсинусів, відносно сторін та кутів сферичних трикутників [27].

Перша таблиця гаверсинусів була опублікована Джеймсом Ендрю у 1805 р.:

$$a = \sin^2(\frac{|\phi_1 - \phi_2|}{2}) + \cos\phi_1 \cos\phi_2 \sin^2(\frac{|\lambda_1 - \lambda_2|}{2}),$$

$$c = 2\arctan(\sqrt{\frac{a}{1-a}}),$$

$$d = Rc$$

де φ – широта, λ – довгота, R – радіус Землі (середній = 6,371 км). d – шукана відстань.

Висотою польоту при обчислюванні відстані буде знехтувано.

Виконання перельоту з точки i у точку j апаратом l описуватиметься змінними x_{ijl}:

$$x_{ijl} = \begin{cases} 0, \textit{ переліт не виконується} \\ 1, \textit{ переліт виконується} \end{cases}, \; i, j = \overline{1, n+k}, \; l = \overline{1, m}. \qquad (2.1)$$

Загальна відстань D, яку мають подолати всі БПЛА під час виконання завдання згідно з отриманим планом операції обчислюється наступним чином:

$$D = \sum_{i=1}^{n+k} \sum_{j=1}^{n+k} \sum_{l=1}^{m} x_{ijl} C_{ij} \, . \qquad (2.2)$$

Загальний час виконання завдання T згідно з отриманим планом операції обчислюється згідно з формулою:

$$T = \max_{l=1,m} \frac{1}{m_{l1}} \sum_{i=1}^{n+k} \sum_{j=1}^{n+k} \sum_{l=1}^{m} x_{ijl} C_{ij} \, , \qquad (2.3)$$

де m_{l1} – згадана вище швидкість l-го БПЛА.

Обмеження. Довжина шляху кожного БПЛА не має перевищувати його максимально допустимої відстані перельоту:

$$\sum_{i=1}^{n+k} \sum_{j=1}^{n+k} x_{ijl} C_{ij} \le m_{l0}, \; l = \overline{1, m}. \qquad (2.4)$$

Кожен БПЛА відвідує певну кількість цілей, отже, прибуття та відправлення від кожної цілі має здійснитися лише один раз. Також, кожна ціль має бути відвідана строго одним БПЛА. Отже, для перевірки коректності розв’язку з точки зору обходу цілей, введемо наступні обмеження.

Один і тільки один БПЛА може покинути ціль:

$$\sum_{l=1}^{m}\sum_{i=1}^{n+k} x_{ijl}C_{ij} = 1,\ \ j = \overline{n+1, n+k}. \qquad (2.5)$$

Один і тільки один БПЛА може прибути до цілі:

$$\sum_{l=1}^{m}\sum_{j=1}^{n+k} x_{ijl}C_{ij} = 1,\ \ i = \overline{n+1, n+k}. \qquad (2.6)$$

Слід зазначити, що у формулах (2.5) та (2.6) при перевірці кількості перельотів, до цілі та від цілі відповідно, переглядаються також перельоти між депо та цілями. Проте перевірка за цими формулами для самих депо не відбувається.

Таким чином, виконується вимога відвідування кожної цілі. Наступна група обмежень пов’язана з вильотами з депо. Необхідно забезпечити, щоб кожен з БПЛА, що має відвідати набір цілей, виконав переліт від одного з депо до цілі, яка не має вхідних перельотів від інших цілей.

Означимо M як множину індексів апаратів, що мають призначені цілі у отриманому плані.

Умова вильоту БПЛА з депо:

$$\sum_{i=1}^{n}\sum_{j=n+1}^{n+k} x_{ijl} = 1,\ \ l \in M. \qquad (2.7)$$

Умова повернення у депо:

$$\sum_{i=1}^{n}\sum_{j=n+1}^{n+k} x_{jil} = 1,\ \ l \in M. \qquad (2.8)$$

Виконання даних умов забезпечує перевірку коректності маршруту кожного БПЛА. Якщо в отриманому плані виконання завдання наявні апарати без призначених цілей, обмеження (2.7) та (2.8) не мають сенсу, тому перевірка відбувається для індексів з множини M.

3 3 РОЗРОБКА АЛГОРИТМІВ МАРШРУТИЗАЦІЇ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ

3.1 Запропоновані методи та алгоритми

При розробці алгоритму для планування операції обльоту цілей з урахуванням наведених обмежень (2.4) – (2.8) слід урахувати наступні особливості:

- оскільки вильоти можуть здійснюватися з різних депо, окрім задачі маршрутизації БПЛА слід також визначити їх початкове розміщення;
- у депо можливе проведення заміни блоків живлення БПЛА під час виконання операції, тож після посадки і заміни запас ходу БПЛА може бути відновлено до початкового рівня, а обліт продовжено;
- депо, з якого здійснюється виліт БПЛА, може відрізнятися від того, у якому буде здійснена його посадка;
- кожне депо незалежно від інших може бути налаштовано таким чином, щоб мати дозвіл лише на здійснення відправлень чи посадки;
- можливі прямі перельоти між депо з метою заміни блоків живлення та подальшого обльоту цілей.

Також необхідно визначити початкове розміщення БПЛА у депо, оскільки це має значний вплив на довжину і тривалість польотів. В окремих випадках, за певних початкових розміщень деякі цілі не можуть бути відвідані взагалі через обмеження на дальність польоту.

В рамках даної роботи було розроблено метод, що дозволяє зробити вибір такого початкового розміщення частиною задачі комбінаторної оптимізації, наведеної у (2.1) – (2.8).

Перш ніж перейти до розробки алгоритму, потрібно ввести низку понять та позначень.

Будуть використані наступні позначення:

$||X||$ – потужність довільної множини X;

$O(x)$ –довільний окіл точки $x \in X$.

За досліджуваний простір візьмемо вибрану множину (назвемо її твірною), між елементами якої існують певні співвідношення. Булеаном 2^X довільної множини X називається множина всіх її підмножин. Нехай на множині X тим чи іншим чином уведена система околів O, тобто для довільного $x \in X$ визначена сім'я множин:

$$o^{\sigma}(x) \subseteq 2^X, \sigma \in I, \qquad (3.1)$$

де I - множина індексів околів.

Дискретний простір - це такий простір, у якому твірна множина X є скінченною ($\| X \| < \infty$) або нескінченною без граничних точок.

Граничні точки ще називають точками конденсації, дотику або накопичення.

Найчастіше використовуються такі види околів:

- метричні;
- топологічні;
- алгоритмічні чи дескриптивні.

Перші два види добре відомі в математиці. Утворення алгоритмічних околів описується певною процедурою, наприклад 2-заміни в задачі комівояжера.

Дескриптивні визначаються шляхом задання в аналітичній чи описовій формі (наприклад шляхом переліку для кожного $x \in X$ елементів, які належать його околам).

Нехай тепер X – метричний простір, тобто на ньому введено метрику $d(x,y)$.

Під (замкненим) метричним околом із заданим радіусом $\rho > 0$ розуміють множину:

$$L_\rho(x) = \{y \in X : d(x,y) \le \rho\}. \quad (3.1)$$

Простір *(X, O)* називається дискретним, якщо:

$$\forall x \in X \exists o^\sigma(x) \in O : o^\sigma(x) = \{x\}. \quad (3.2)$$

Таким чином, для кожної точки дискретного простору існує окіл, який складається лише із цієї точки – саме в такому разі точку називають ізольованою.

Простір *X* називається дискретним, якщо він складається лише із ізольованих точок.

Варіант розв'язку $x_* \in S$ називається субоптимальним розв’язком задачі, якщо:

$$f(x_*) \le f(y), \forall y \in S. \quad (3.3)$$

Якщо *S* збігається з околом точки x, тобто $S = o^\sigma(x)$, де $o^\sigma(x)$ – деякий окіл точки *x*, то такий субоптимальний розв'язок називається локально оптимальним, або локальним.

Уведені означення дозволяють формально визначити поняття глобального екстремуму.

Якщо умова (3.3) виконується для $S = X$, то точка x_* називається глобальним, або точним, розв’язком задачі. Якщо система околів така, що будь-який локально оптимальний розв'язок задачі глобальний, то кажуть, що така система околів є точною.

Часто під ЗКО розуміють проблему пошуку екстремумів за даної цільової функції, коли *X* – комбінаторний простір.

Комбінаторним простором вважають сукупність комбінаторних об'єктів певного типу, утворених із елементів заданої скінченної множини (твірна множина), хоча при цьому формального означення не наводять.

Базисними околами довільної точки $x \in X$ будемо називати наступну множину:

$$B_x = \{o^\tau(x) \in O : \left\|o^\tau(x)\right\| > 1 \,\&\, \nexists \gamma : 1 < \left\|o^\gamma(x)\right\| < \left\|o^\tau(x)\right\|\} \qquad (3.4)$$

Комбінаторним простором назвемо дискретний локально скінченний у комбінаторному розумінні простір, який має не більш ніж зліченну кількість елементів.

Задача називається задачею комбінаторної оптимізації (ЗКО), якщо простір її розв’язків X є комбінаторним.

При дослідженні складності алгоритмів задач оптимізації останні розглядаються як задачі розпізнавання вигляду: для даної індивідуальної ЗКО і цілого числа M визначити, чи існує такий припустимий розв'язок $x \in D \subseteq X$, що $f(x) \le M$.

Зрозуміло, що для задачі максимізації нерівність мала б ви вигляд $f(x) \ge M$. Можна показати, що ця форма постановки задач має таку саму трудомісткість, як і відповідний оптимізаційний варіант.

Розрізняють два класи проблем: P та NP. Ці класи можна визначити точніше за допомогою будь-якого формального означення алгоритмів, такого, наприклад, як машина Тьюрінга. Не заглиблюючись у більш формальний опис, надалі нам буде достатньо означити їх так.

Клас P складають задачі, для розв'язування яких відомі алгоритми з поліноміальною складністю (polynomial-time algorithms).

До поліноміальних відносять алгоритми, складність (трудомісткість) яких обмежена поліномом, степінь якого залежить від розміру входу ЗКО. Отже, це клас відносно простих задач, для яких існують ефективні алгоритми.

До класу NP (nondeterministic polynomial), який здається ширшим, належать задачі, що можуть бути розв'язані недетермінованим поліноміальним алгоритмом.

Такі алгоритми мають дві фази:

– на першій знаходять припустимий варіант розв'язку;
– на другій цей варіант перевіряється детермінованим поліноміальним алгоритмом верифікації.

Задача комбінаторної оптимізації n є NP-повною (NP-complete), якщо виконуються дві умови:

– $n \in NP$;
– усі задачі з NP можуть бути зведені до n за допомогою поліноміального алгоритму.

На практиці для верифікації зазвичай лише показують, що деяку відому NP-повну задачу можна поліноміально перетворити на досліджувану задачу. Інколи можна показати, що всі задачі з NP поліноміально зводяться до деякої задачі n, але не вдається довести, що $n \in NP$. Такі задачі відносять до NP-складних.

Для потреб практики важливо, щоб трудомісткість алгоритмів була обмежена поліномом від вхідних даних задачі. Нині предметом дискусій є питання, чи $P = NP$. У разі негативної відповіді теоретично не існує поліноміального алгоритму, тому при розв'язуванні таких задач одразу слід зосередитися на розробці потужних наближених алгоритмів (рисунок 3.1), які, хоча й не гарантують отримання оптимального розв'язку, здатні знаходити оптимальний чи близький до нього розв'язок за прийнятний час [2].

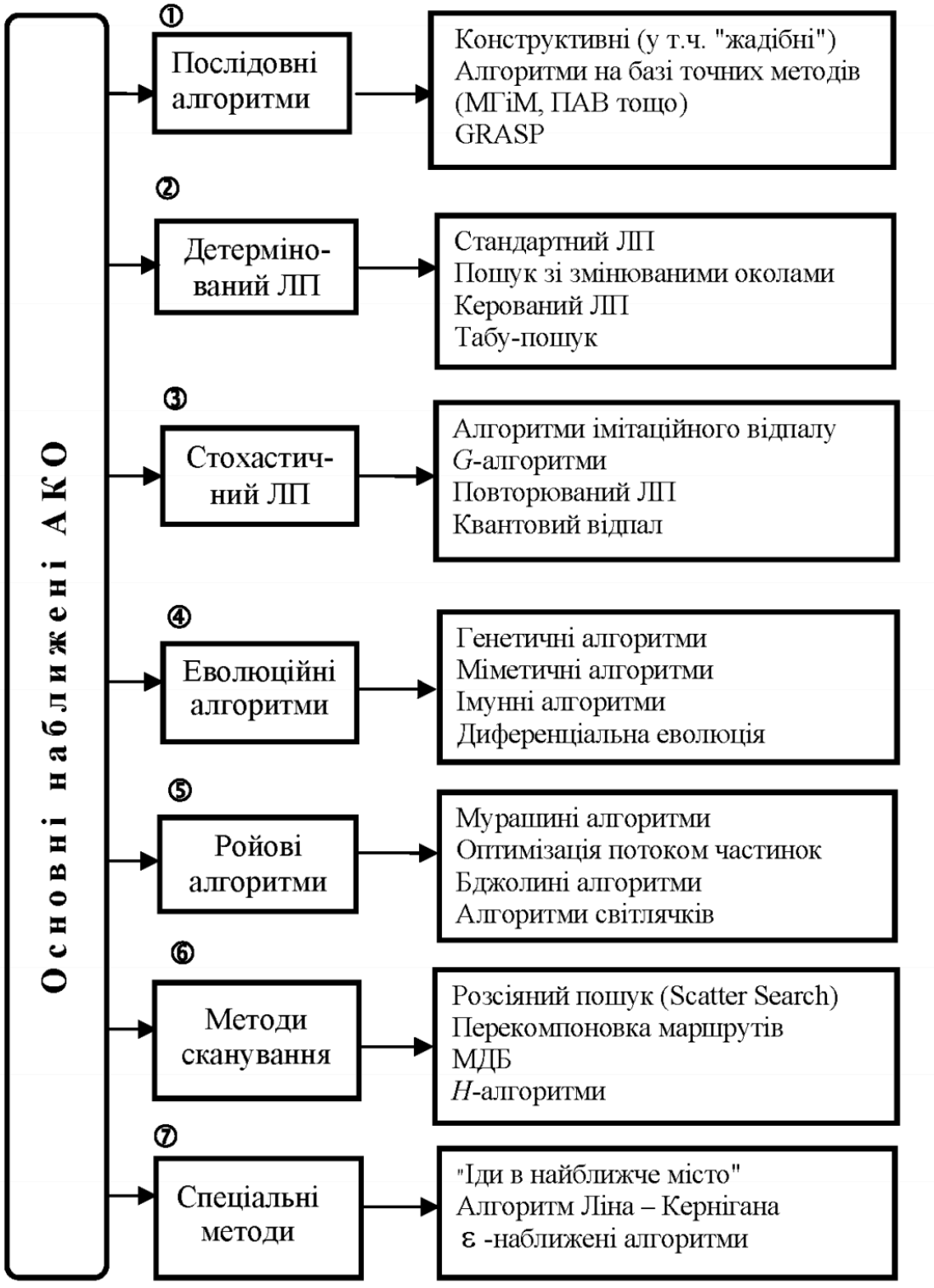


Рисунок 3.1 – класифікація основних наближених методів комбінаторної оптимізації [2]

Викладене справедливе не лише для ЗКО, які розглядалися у формі розпізнавання, а й для низки інших проблем, що формулюються відразу як задачі розпізнавання (наприклад задача виконуваності булевої форми чи задача про гамільтонів цикл).

3.2 Класифікація алгоритмів комбінаторної оптимізації

Класифікація алгоритмів комбінаторної оптимізації (АКО) за отримуваним розв'язком. Для простоти викладу поки вважатимемо, що $D = X$, тобто розглянемо ЗКО без обмежень: необхідно знайти елемент $x_* \in X$ такий, що:

$$x_* = \arg \underset{x \in X}{ext} f(x). \tag{3.5}$$

Будемо вважати, що АКО - певна процедура A, яка переводить задану підмножину $Z \subset X$ (початкові наближення) у множину $X_* \subset X$:

$$AZ = X_*, \tag{3.6}$$

тобто X_* – це множина знайдених розв’язків задачі.

Існує багато підходів до класифікації АКО - за точністю, типом використаних просторів, структурою обчислювальної схеми тощо (рисунок 3.2).

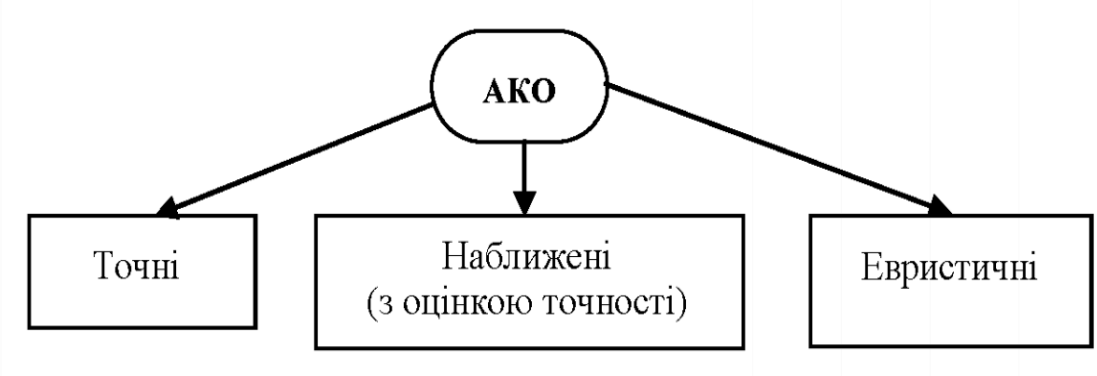


Рисунок 3.2 – класифікація алгоритмів комбінаторної оптимізації за точністю [2].

Точні АКО – це такі методи, які знаходять глобальний розв’язок, тобто для них $X_* \subseteq \arg ext$. Наближені АКО діляться на алгоритми з апріорною та апостеріорною оцінками точності.

Евристичні алгоритми будуються на основі правдоподібних міркувань (наприклад, алгоритм "іди в найближче місто", що використовується для розв'язання задачі комівояжера), але не в змозі оцінити точність знайденого розв'язку. Часто дослідники називають наближеними як алгоритми з оцінкою точності, так і евристичні.

3.3 Класифікація АКО за типом обчислювальної схеми.

Наближені алгоритми за типом обчислювальної схеми прийнято ділити на конструктивні та ітераційні.

Нехай маємо певну множину $Y \supseteq X$.

Конструктивні алгоритми (інші назви - прямі, послідовні) - це такі алгоритми, у яких $Y \supset X.(Y \neq X)$, тобто вони оперують у просторі, що є розширенням простору розв’язків *X*.

Починаючи "з нуля" чи якогось фрагмента розв'язку, вони поступово формують повний розв'язок.

Приклади конструктивних алгоритмів для різних ЗКО:

- задача комівояжера – алгоритм "іди в найближче місто", який на кожному наступному кроці додає до маршруту вершину, відстань до якої є найменшою з усіх можливих;
- квадратична задача про призначення – евристика, відповідно до якої об'єкти (елементи) з інтенсивнішими потоками розміщуються в першу чергу;
- задача про мінімальне кістякове дерево – евристика, відповідно до якої на кожному наступному кроці в побудований фрагмент

кістякового дерева включається та вершина, яка мінімально збільшує його сумарну довжину й не утворює підцикл.

Ітераційні алгоритми – це такі алгоритми, які на кожному кроці опрацьовують "повні" розв'язки, тобто для них простір пошуку $Y \equiv X$. Починаючи з деякого $x^o \in X$, ітераційні алгоритми намагаються його поліпшувати покроково:

$$x^{(h+1)} = A^{(h)} x^{(h)}, h = 0,1,...,$$

де $A^{(h)}$ – ітераційна процедура, яка у більшості випадків не залежить від кроку *h*, тобто $A^{(h)} = A$. Ітераційні методи, які оперують на кожному кроці одним (поточним) розв'язком, називаються траєкторними.

Інколи в зарубіжній літературі такі методи називаються базованими на одному розв'язку чи стані (Single-Solution Based/Single-State Methods), а під траєкторними розуміють такий їх підклас, який породжує послідовність сусідніх розв'язків – траєкторію у просторі пошуку.

Уникаючи певних термінологічних ускладнень, будемо всі такі методи називати траєкторними. Алгоритми, які опрацьовують на кожній ітерації не один, а кілька розв'язків одночасно, називаються популяційними (Population-Based Methods).

Отже, для траєкторних алгоритмів $||Z|| = ||X_*|| = 1$, а для популяційних – $||Z|| > 1, ||X_*|| > 1$.

За складністю структури АКО можна виділити:

- прості алгоритми;
- комбіновані алгоритми;
- метаевристики;
- гібридні метаевристики;
- гіперевристики.

Комбіновані алгоритми утворюються шляхом послідовного застосування двох чи більше ітераційних алгоритмів з передаванням розв’язків від одного до іншого. У метаевристиках здійснюється вкладення одного алгоритму / процедури в іншу стратегію.

Гіперевристикою (гіперевристичним алгоритмом) називають метод пошуку, орієнтований на автоматизацію процесів вибору, комбінування або адаптації чи налаштування кількох простіших алгоритмів (евристик або метаевристик) для ефективного розв'я зання ЗКО чи їх класів.

Це може досягатися як вибором наявних евристик чи їх фрагментів, так і генеруванням нових. Таким чином, якщо метаевристики та інші алгоритми здійснюють переважно пошук у просторі розв’язків ЗКО, то простором пошуку для гіперевристик є множина евристик (простіших алгоритмів чи їх частин).

За впливом на ландшафт пошуку більшість АКО можна віднести до таких, що залишають його незмінним. Проте, є алгоритми, які модифікують цей ландшафт шляхом:

- зміни простору розв'язків (наприклад, послідовні алгоритми);
- зміни цільової чи оцінкової функції (алгоритми керованого локального пошуку);
- варіації системи околів, що використовується при пошуку (алгоритми локального пошуку (ЛП) зі змінними околами, метод вектора спаду з пульсуючими околами).

Якщо робота алгоритму базується на безпосередніх даних ЗКО, то такі АКО належать до задаче-орієнтованих алгоритмів. У деяких нових АКО використовуються не стільки прямі дані ЗКО, скільки спеціальна модель задачі, що розв'язується (наприклад феромонна матриця та матриця маршрутів у алгоритмі оптимізації мурашиною колонією), – такі алгоритми отримали назву моделе-орієнтованих.

Для розв’язування сформульованої задачі (2.1) – (2.8) розроблено наближений алгоритм оптимізації мурашиною колонією (ОМК) з використанням детермінованого локального пошуку (ДЛП) в якості вбудованої процедури. Також для формування початкового допустимого розв’язку розроблено жадібний алгоритм.

3.4 Ключові аспекти реалізованого жадібного алгоритму

У рамках розробленого жадібного алгоритму кожен БПЛА розглядається як самостійний агент, що характеризується:

- початковим запасом ходу;
- залишковим запасом ходу (відстань, яку БПЛА ще може пролетіти, починаючи з поточного стану, рівний початковому при вильоті з депо);
- пройденим шляхом;
- поточною позицією (одна з вершин).

На початку роботи алгоритму всі БПЛА розміщуються у «нульовій» точці – умовна точка, однокрокові перельоти з якої безпосередньо у депо заборонені, а відстань до цілі визначається як мінімальна відстань від даної цілі до депо.

Такий підхід дозволяє зробити задачу початкового розміщення БПЛА частиною задачі їх маршрутизації. Отже, перехід БПЛА k в один крок з «нульової» точки o до цілі i буде здійснюватися наступним чином:

- у якості позиції БПЛА встановлюється ціль i;
- у шлях БПЛА додається найближче до цілі депо та сама ціль;
- від залишкового запасу ходу віднімається найменша з відстаней від цілі до депо;
- ціль помічається відвіданою.

Переходи між цілями та депо (i – початкова точка, j – кінцева) здійснюються за аналогічною схемою, проте без проміжних точок:

- у якості позиції БПЛА встановлюється точка *j*;
- у шлях БПЛА додається точка *j*;
- від залишкового запасу ходу віднімається відстань між точками *i* та *j;*
- якщо *j* – ціль, позначається відвіданою.

У разі переходу в депо здійснюється заміна блоків живлення, залишковий запас ходу встановлюється на рівні початкового.

Обчислювальну схему розробленого жадібного алгоритму можна подати наступним чином:

```
procedure greedy (x)
  ініціалізація_алоритму;
  while поточний_стан ≠ повний_розв'язок do
    A = список_допустимих_переходів;
    foreach БПЛА do
      A+=допустимі_переход;
    n = найкоротший_перехід(A);
    перейти_в_наступний_стан(n);
  endwhile
завершити діяльність;
end
```

Рисунок 3.3 – обчислювальна схема жадібного алгоритму маршрутизації БПЛА

Слід зазначити також, що при здійсненні переходів слід перевіряти досяжність інших вершин після його здійснення, інакше можлива ситуація, за якої допустимий розв’язок не буде знайдено.

3.5 Загальна схема алгоритму детермінованого локального пошуку

Алгоритми детермінованого локального пошуку - це сім'я ітераційних методів, заснована на частковому перебиранні варіантів на кожній ітерації серед точок околу поточної точки, тобто серед сусідніх до неї. В алгоритмах цього типу замість повного перебору застосовується спрямований локальний перебір у підмножинах варіантів, які називаються околами. Цим пояснюється їх назва – алгоритми локального пошуку (Local search). У сфері КО алгоритми ЛП мають давню історію через свою наочність і високу ефективність. Наприклад, перший алгоритм локального пошуку для задачі комівояжера був запропонований ще в 1956 р., а локальний пошук для задачі розміщення обладнання був розроблений у 1962 р.

Загальна схема ЛП у задачах мінімізації виглядає наступним чином. Починаючи з деякого припустимого розв'язку задачі, новий розв'язок із кращим значенням цільової функції шукають у його околі. Якщо такий розв'язок знайдено, то він приймається і пошук поліпшення розв'язку далі здійснюється вже в його околі і т. д. Алгоритм закінчується, коли досягнуто локального оптимуму, тобто коли в околі поточного розв'язку немає ніякого іншого варіанта з меншим значенням цільової функції.

Загальна схема алгоритмів детермінованого локального пошуку може бути подана таким чином:

- генерація початкового припустимого розв'язку x, який обираємо як поточний варіант;

- чергова ітерація – формуємо окіл *O(x)* поточного варіанта й точно чи наближено знаходимо елемент $y \in O(x)$, який є субоптимальним розв'язком у цьому околі. Якщо $y \neq x$, то знайдений елемент оголошується черговим поточним варіантом і починається чергова ітерація, інакше – наступний пункт;
- завершення роботи алгоритму: x – локальний розв'язок, якщо на останній ітерації здійснюється вичерпний пошук в околі.

3.6 Ключові аспекти реалізованого алгоритму ДЛП

Загалом, принциповими моментами реалізації конкретних алгоритмів локального пошуку є:

- визначення околів *O(x)*;
- генерація чергової точки $y \in O(x)$;
- критерій завершення перегляду точок у поточному околі та переходу до наступного;
- спосіб обчислення величини зміни цільової функції при переході до нового поточного варіанта;
- критерій завершення;
- формування початкового наближення.

Використано відомий алгоритм 2-замін Ліна – один з сімейства алгоритмів k-замін (k-exchange), що базуються на процедурі вилучення k ребер і додавання к нових. Для побудови сусідніх варіантів (околу) до поточного маршруту було запропоновано використовувати такі процедури:

- транспозиції міст;
- вставка міста;
- переміщення фрагмента шляху;

– заміна *k* ребер.

Схема алгоритму 2-opt Ліна, що базується на процедурі 2-заміни, поданій рисунком 3.4.

```
procedure 2-OPT(x);
   створення початкового маршруту x за допомогою деякого
   алгоритму;
   while не виконуються умови завершення do
      for i = 1 to n do
         for j = 1 to n do
            вибрати несуміжні ребра e_i та e_j;
             if існують ребра e_k, e_l такі, що забезпечують меншу
              довжину маршруту then
                 замінити в x ребра e_i та e_j на ребра e_k та e_l;
         endfor
      endfor
   endwhile
   return x
end
```

Рисунок 3.4 – псевдокод алгоритму 2-opt [2]

Обчислювальна складність алгоритму – $O(n^2)$.

Аналогічно записується алгоритм 3-замін, у якому вилучаються три ребра з наявного розв'язку та вставляються три нові так, щоб знову був гамільтонів цикл. Така процедура підвищує ефективність пошуку, але суттєво збільшує час розрахунків. Обчислювальна складність алгоритму - $O(n^3)$. За результатами досліджень, алгоритм 3-opt може знаходити маршрути в деяких евклідових задачах комівояжера на площині, довжина

яких лише на 2,5-4 % більша від оптимальної. У своїх роботах Лін показав, що 3-заміни значно кращі за 2-заміни, а ефект від використання 4-замін є незначним і нівелюється суттєвим зростанням обчислювальних витрат. Саме через високу обчислювальну складність у даній роботі вирішено відмовитися від 3-замін і скористатися 2-замінами. Оскільки алгоритм ДЛП використовується у якості демону для нового розробленого алгоритму, обчислення виконуються багато разів і мають бути якомога швидшими [9].

3.7 Загальна схема алгоритму ОМК.

Серед відомих алгоритмів ройового інтелекту (swarm intelligence) в комбінаторній оптимізації значного поширення набули алгоритми ОМК (ant colony optimization – ACO) [8]. Вони успішно застосовуються для розв’язування багатьох типів задач комбінаторної оптимізації, починаючи з класичної задачі комівояжера.

Основні компоненти обчислювальної схеми мурашиних алгоритмів такі: модель задачі, що подається спеціальним графом; феромонні значення; евристична інформація; пам'ять (локальна та глобальна). В алгоритмах ОМК формується спеціальна модель задачі, що розв'язується, тому вони належать до класу моделе-орієнтованих методів. Модель задачі маршрутизації БПЛА подається у вигляді зваженого графа $G(V,E)$, де $v_i \in V,\ i = 1,...,n+k$ – вершини, що відповідають компонентам розв'язку, а $e_{ij} \in E,\ e_{ij} = (v_i, v_j),\ v_i, v_j \in V$ – ребра, які відповідають можливим з'єднанням (переходам) між відповідними вершинами. Для кожного ребра визначена функція вартості з'єднання, що відповідає відстані по поверхні між вершинами, з’єднаними даним ребром, якщо одна з вершин – ціль, інша – ціль чи депо або ж нескінченності, якщо обидві вершини є депо.

На кожному кроці алгоритму для будь-якої вершини $i \in V$ може бути побудована множина сусідніх вершин N_i.

Умови задачі, визначають набір обмежень $\Pi = \Pi(V, E, t)$ для елементів V та E, описані формулами (4)–(8), які визначають припустимість зав'язків між компонентами та з'єднаннями, а в підсумку – і побудованого з них розв'язку. Розв'язки задачі подаються як припустимі шляхи на графі G.

Евристична інформація η_{ij} – це числове значення, що не залежить від знайдених на попередніх кроках розв'язків і відображає ступінь бажаності включення в побудований фрагмент розв'язку того чи іншого нового ребра графа моделі $e_{ij} \in E$. Евристичні значення η_{ij} базуються на апріорній інформації, що відображає умови конкретної задачі та надається джерелом, відмінним від мурах.

Рівень феромону (феромонний слід) – τ_{ij}, що відповідає ребру $e_{ij} \in E$, – це додатне число, яке показує, наскільки часто мурахами використовувалося це ребро на попередніх кроках чи при формуванні повного розв'язку. Феромонні сліди виконують роль довготривалої пам'яті для мурах [6].

Загальну обчислювальну схему алгоритмів ОМК можна подати як на рисунку 3.5.

```
procedure ACO (x)
    ініціалізація_алоритму;
    while критерій_завершення_не_задоволений do
    формування_популяції_мурах;     {поточне покоління}
    foreach мураха_з_популяції do    {життєвий цикл мурахи}
    ініціалізація_мурахи;
    M = оновлення_пам'яті_мурахи;
    while поточний_стан ≠ повний_розв'язок do
        A=локальна_ матриця_мурашиних_маршрутів;
        сформувати_множину припустимих_вершин;                (17)
        p=обчислити_ймовірність_переходів(A, M, П);
        наступний_стан=правило_прийняття_рішення(p, П);
        перейти_в_наступний_стан(наступний_стан);
        if онлайнове_покрокове_оновлення_феромону then
            відкласти_феромон_на_відвіданій_дузі;
            поновити_ матрицю_мурашиних_маршрутів_A;
        endif
        M =оновити_внутрішній_стан;
 endwhile
 if онлайнове_відстрочене_оновлення_феромону then
    foreach відвіданої_дуги_побудованого_розв'язку do
        відкласти_феромон_на_відвіданій_дузі;
        оновити_матрицю_мурашиних_маршрутів_A;
     endforeach
 endif
 завершити діяльність;
 endforeach
 випаровування_феромону_та_оновлення_рекорду (x);
 дії_Демона;              {необов'язково}                      (18)
 endwhile
end
```

Рисунок 3.5 – обчислювальна схема алгоритмів ОМК [2].

3.8 Особливості нового розробленого алгоритму

У розробленому алгоритмі в якості множини припустимих вершин (17) обрано множину невідвіданих вершин, до яких одна з мурах зі своєї поточної позиції може дістатися не більше, ніж за два кроки [6]. У якості дій Демона (18) виступає оптимізація отриманого розв'язку за допомогою алгоритму ДЛП.

Спочатку для кожної мурахи k із популяції здійснюється формування підмножини припустимих вершин $N_i^k \subseteq N_i$ із множини вершин, сусідніх для тієї вершини $i \in V$, яка є останньою в поточному фрагменті маршруту. Отже, N_i^k є множиною можливих переходів за один крок для мурахи k з вершини i. Потім додатково будується D_i^k – множина точок, до яких можливий перехід у два кроки, тобто, всі невідвідані точки з можливістю прямого переходу від однієї з точок попередньої множини N_i^k:

$$D_i^k = \bigcup_{s \in N_i^k} N_s{}^k .$$

Іншими словами, множина D_i^k є об'єднанням множин допустимих переходів для кожної з точок s з множини допустимих переходів для обраної вершини i та мурахи k.

Перехід k-ї мурахи з вершини i в j через вершину s на поточній ітерації t здійснюється з ймовірністю, що розраховується за наступною формулою:

$$p_{isj}{}^k = \frac{[\tau_{is}(t) \cdot \tau_{sj}(t)]^{\alpha} \cdot [\eta_{is}(t) \cdot \eta_{sj}(t)]^{\beta}}{\sum\limits_{r \in N_i^k} [\tau_{ir}(t)]^{\alpha} \cdot [\eta_{ir}(t)]^{\beta} \cdot \sum\limits_{r \in N_s{}^k} [\tau_{rj}(t)]^{\alpha} \cdot [\eta_{rj}(t)]^{\beta}}, \qquad (19)$$

де $N_{ilj}{}^k = \{l : l \in N_i^k,\ j \in N_l{}^k\}$.

Формула (19) описує ймовірності для переходів через проміжну вершину, тобто, двокрокового алгоритму ОМК. Для прямих переходів до вершин з множини N_i^k застосовується наступна формула:

$$q_{ij}{}^k = \frac{[\tau_{ij}^{\alpha}(t) \cdot \eta_{ij}^{b}(t)]}{\sum_{r \in N_i^k} [\tau_{ir}^{\alpha}(t) \cdot \eta_{ir}^{b}(t)]}. \qquad (20)$$

Вибір точки для наступного переходу здійснюється серед елементів множини $D_i^k \cup N_i^k$, тобто, може бути обрано як прямий, так і двокроковий перехід. Ймовірність кожного переходу для мурахи k з вершини і обчислюється як $p_{isj}{}^k$, $s \in N_i^k$, $j \in D_i^k$ для двокрокових переходів та $q_{ij}{}^k$, $j \in N_i^k$ для прямих переходів.

У якості демона (18) використано описаний вище алгоритм 2-замін Ліна [9].

4 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1 Використані технології та архітектурні рішення

Складність та недоцільність встановлення програмного комплексу на персональні комп'ютери зумовлена, перш за все, наступними чинниками:

- відмінностями програмного забезпечення на різних комп'ютерах – різні операційні системи, компілятори, інтерпретатори, архітектура тощо;
- потребою у встановленні та налаштуванні програмного комплексу користувачем, що не володіє інформацією про реалізацію програми;
- обмеженою масштабованістю та обчислювальною спроможністю локального серверу.

Це зумовлює потребу у розміщенні програми на віддаленому сервері та налагодження інтерфейсу взаємодії з нею. У якості архітектурного рішення для виділення обчислювальної частини програмного комплексу було обрано мікросервіси – архітектурний стиль, в якому складні застосування, створені як сукупність маленьких, легких, самодостатніх, незалежних, слабко зв'язаних сервісів, кожен з яких відповідальний за конкретний процес. Такий стиль протиставляється монолітному стилю, згідно з яким додатки будуються як єдине ціле.

Мікросервіси пов'язуються один з одним на основі потреби виконання певної послідовності дій. Обмін даними здійснюється через програмний інтерфейс застосування (Application Programming Interface, API), для якого не має значення мова програмування.

Такий підхід схожий на командну роботу – кожен учасник сконцентрований на окремій задачі. Йому дають відповідальність, свободу

і довіру щодо виконання своєї роботи найкращим чином. Основні переваги мікросервісної архітектури описані далі.

Швидке внесення змін та легке, безпроблемне розгортання – оскільки кожен мікросервіс розгортається та змінюється окремо. Тож якщо відбуваються зміни в одному з них, зміни можуть бути розгорнуті, без повторного запуску інших мікросервісів, які можуть продовжувати працювати. Можна вносити будь-які зміни настільки часто, наскільки потрібно, щоб застосування завжди відповідало вимогам. В монолітній архітектурі все інакше – будь-яка зміна може потребувати розгортання цілої складної системи з самого початку.

Будь-який мікросервіс у системі можна замінити. Його можна переписати з нуля окремо від систами загалом, в межах прийнятного часу та бюджету.

Якщо ж ідеться про монолітне застосування, його також можна модернізувати, уникнувши докорінної перебудови, якщо звести його до мікросервісної архітектури.

Потенційно мікросервіси легші для розуміння, підтримки і тестування. Мікросервіси зазвичай невеликі за обсягом коду, з відокремленною логікою. Завдяки цьому, командам розробників легше їх розуміти і підтримувати. Їх також легше повністю покривати автоматизованими тестами.

В будь-якому випадку, тестувальникам потрібно перевіряти лише невеликий незалежний фрагмент, тож процес перевірки та отримання зворотнього зв’язку отримується швидше.

Неполадки у мікросервісі не повинні вивести з ладу все застосування. До того ж, обробка таких виключних ситуацій спрощується через ізольованість компонентів системи.

Мікросервіси на різних мовах і платформах можуть працювати разом, що є актуальним для розглянутого програмного комплексу. Більш детально використані мови та платформи буде описано далі.

Мікросервіси можуть бути абсолютно різними і при цьому ефективно взаємодіяти у якості одного цілого. Потрібно лише визначити правила комунікації між ними. У випадку мікросервісів комунікація відбувається найчастіше за допомогою HTTP, а також – сервісної шини, бінарного протоколу тощо [28].

Можливість використовувати різні мови програмування та інструменти — це індивідуальний підхід до побудови сервісів. Він дозволяє обирати інструмент для конкретної роботи, а також долучати нові технології при подальшому розвитку [28].

З вищесказаного випливає, що винесення програми планування операції у окремий мікросервіс надає багато переваг при розробці та дозволяє інтегрувати його у різні системи незалежно від їх технологій та платформ.

Вхідні дані мають задаватися без прямого доступу до сервера, втручання у код чи виконувані файли програми. Отже, необхідно розробити інтерфейс вводу та виводу, що дозволить давати програмному комплексу завдання і отримувати результат їх виконання.

Для взаємодії з іншими компонентами системи використано API, що дозволяє здійснити обмін інформацією зі сторонніми системами.

API — application programming interface (інтерфейс програмування додатків), набір правил і механізмів, за допомогою яких один додаток або компонент взаємодіє з іншими. Далі наведено деякі з найрозповсюдженіших способів реалізації API:

- XML-RPC;
- JSON-RPC;
- SOAP;

– REST.

RPC (remote procedure call — віддалений виклик процедур») – поняття, що поєднує давні і сучасні протоколи, які дозволяють викликати метод в іншому додатку. XML-RPC – протокол, що з'явився в 1998 році незабаром після появи XML. Спочатку він підтримувався Microsoft, але незабаром Microsoft повністю переключилася на SOAP. Незважаючи на це, XML-RPC продовжує існувати і підтримуватися у різних мовах (особливо в PHP) [29].

SOAP також з'явився в 1998 році за підтримки Microsoft. Він був анонсований як революція в світі. Основна критика протоколу викликана його складністю та важкістю. У той же час, були і ті, хто вважав SOAP справжнім проривом. Протокол продовжував розвиватися у вигляді нових специфікацій, поки у 2003 році W3C не затвердила в якості рекомендації SOAP 1.2 [29].

Потім з'явився дійсно простий підхід – REST. Абревіатура REST розшифровується як representational state transfer — «передача стану представлення» або, краще сказати, подання інформації в зручному для форматі. Термін «REST» був введений Роєм Філдінгом у 2000 році. Основна ідея REST в тому, що кожне звернення до сервісу переводить клієнтське застосування у новий стан. По-суті, REST – не протокол і не стандарт, а підхід, архітектурний стиль проектування API [29].

Оскільки взаємодія з планувальником операції є відносно простою і обмежується лише двома діями (відправкою даних задачі та отриманням результату), використано підхід REST API.

У якості формату обміну даними було обрано JSON через інтуїтивну зрозумілість та незначну надлишковість для даних про цілі, депо, БПЛА та їх характеристики.

JSON (JavaScript Object Notation) – текстовий формат обміну даними між комп'ютерами. JSON базується на тексті та може бути без додаткових перетворень прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

JSON будується на двох структурах:

- набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом;
- впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі. Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його саме на цих структурах.

У JSON використовуються наступні форми структур.

Об'єкт — це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.

Масив — це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

Значення може бути:

- рядком в подвійних лапках;
- числом;
- логічними true чи false;
- null;
- об'єктом;

– масивом.

Ці структури можуть бути вкладені одна в одну.

Далі буде наведено формат взаємодії з планувальником операцій, що дозволяє розміщувати завдання, отримувати, обробляти та відображати результат.

4.2 Структура програмного комплексу планувальника операцій

Програмний комплекс складається з наступних частин:

– модуля керування;
– модуля обчислення;
– модуля візуалізації.

Обробка запитів, підготовка даних та обмін інформацією між програмою та зовнішніми сервісами відбувається у модулі керування. Звідти підготовлена інформація про вхідні дані передається до модуля обчислення.

У модулі обчислення відбувається створення плану проведення операції з використанням розробленого алгоритму, результат роботи повертається у модуль керування.

Отриманий план обробляється і передається у модуль візуалізації, де формується мапа з вказанням планованих перельотів та текстовий опис операції. Вся інформація, необхідна для відображення, передається назад, у модуль керування, звідки виводиться користувачу чи системі, що розмістила задачу.

На рисунку 4.1 показана діаграма компонентів системи.

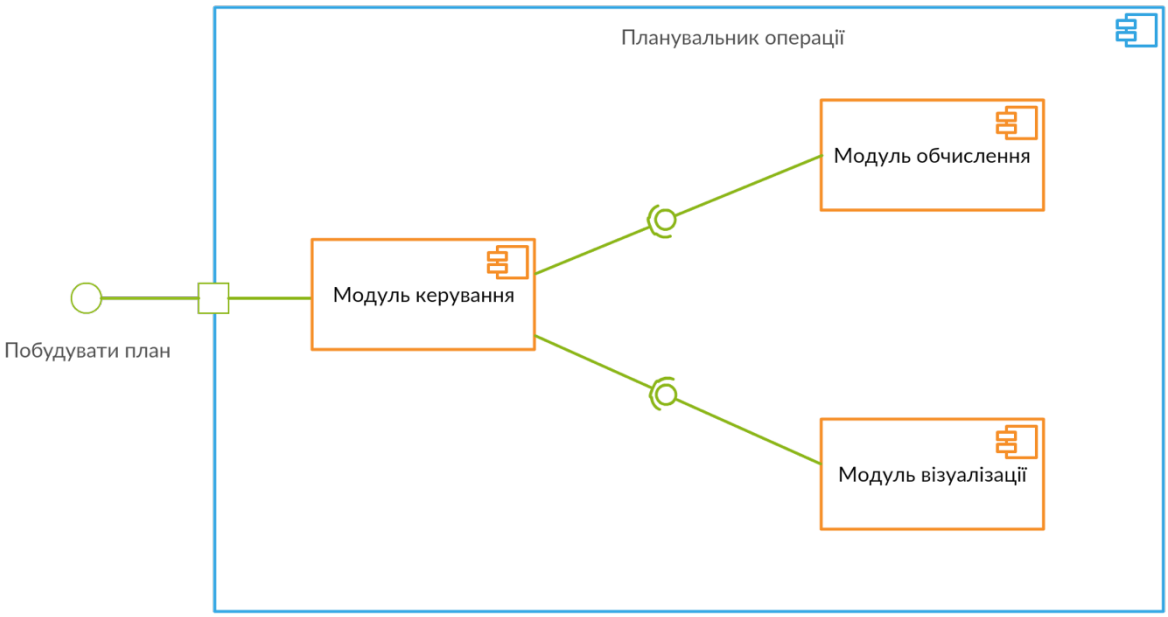


Рисунок 4.1 – діаграма компонентів планувальника операцій

Модуль керування надає зовнішній інтерфейс, який дозволяє поставити завдання програмному комплексу та отримати план проведення операції.

Модуль обчислення отримує інформацію через стандартний потік вводу, відправляє результати через стандартний потік виводу. Модуль керування створює дочірній процес, у якому запущено екземпляр програми для обчислення, та обмінюється з нею інформацією для розв’язання задачі.

Модуль візуалізації надає інтерфейс для відправки матриці усіх точок задачі та визначених шляхів БПЛА, повертає html сторінку з побудованим на OpenStreetMap планом операції.

У даному розділі також надано інструкцію для роботи з програмним комплексом. У прикладах продемонстровано можливий процес інтеграції мікросервісу зі сторонньою системою.

Графічний інтерфейс для введення даних не передбачено, оскільки основна функція мікросервісу – обслуговувати сторонні системи, а не безпосередньо користувачів.

4.3 Інструкція з використання програмного комплексу

Для введення даних, що описують операцію, та отримання результатів програмний комплекс надає REST API. Для зовнішніх запитів відкрита єдина функція – побудувати план операції.

Для того, щоб надіслати задачу в обробку та отримати план операції, потрібно надіслати POST-запит на адресу «/api/v1/mission/compute». Необхідно також вказати заголовок «Content-Type: application/json», а у тілі запиту передати JSON з вхідними даними.

На рисунку 4.2 наведено приклад такого запиту у застосуванні Postman.

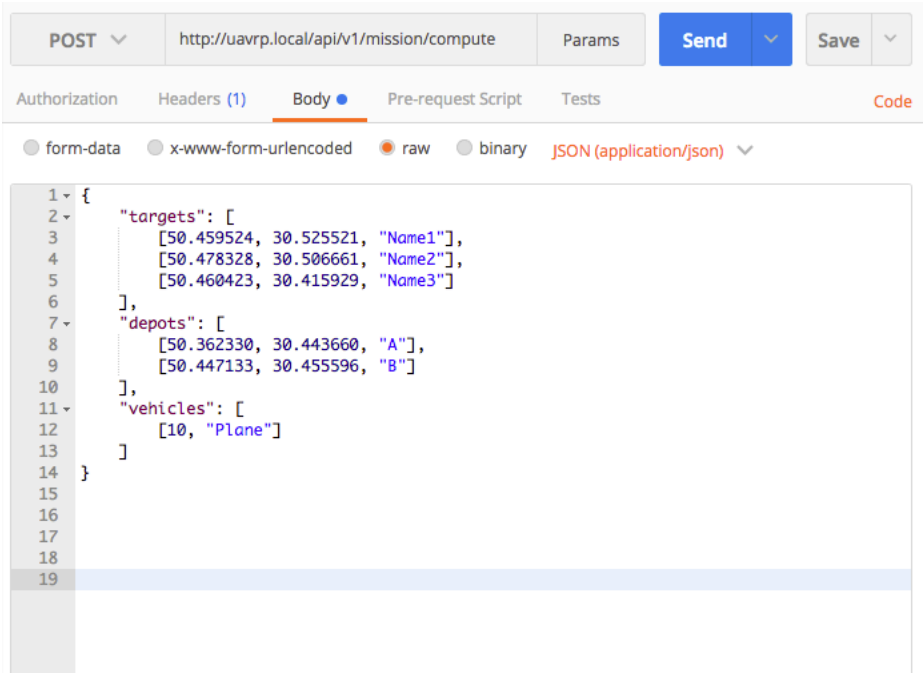


Рисунок 4.2 – приклад запиту за допомогою застосування Postman

На рисунку 4.3 наведено приклад того самого запиту у відокремленому вигляді. Подібні запити до програмного комплексу має виконувати система, що інтегрується з ним. Таким чином, робота з планувальником операцій не потребує зміни його кодової бази чи, тим більше, прямого об’єднання з кодом сторонньої системи.

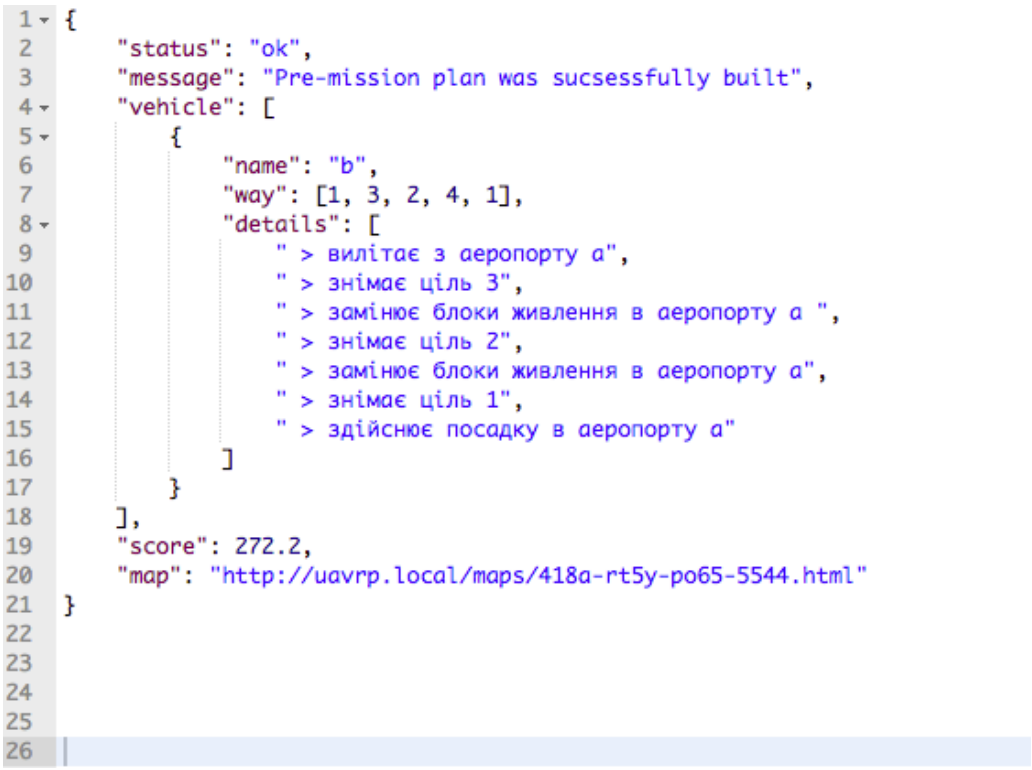
```
POST /api/v1/mission/compute HTTP/1.1
Host: uavrp.local
Content-Type: application/json
Cache-Control: no-cache

{
   "targets": [
            [50.459524, 30.525521, "Name1"],
            [50.478328, 30.506661, "Name2"],
            [50.460423, 30.415929, "Name3"]
     ],
   "depots": [
            [50.362330, 30.443660, "A"],
            [50.447133, 30.455596, "B"]
     ],
   "vehicles": [
     [10, "Plane"]
     ]
}
```

Рисунок 4.3 – код запиту до програмного комплексу

У запиті можуть використовуватися наступні параметри:

- targets – масив, що містить цілі у вигляді масивів з трьох елементів: широта, довгота, ідентифікатор;
- depots – масив, що містить депо у вигляді масивів з трьох елементів: широта, довгота, ідентифікатор;
- vehicles – масив, що містить доступні БПЛА у вигляді масиву з двох елементів: запас ходу та ідентифікатор;
- details – булеве значення, що вказує на те, чи виводити детальну текстову інформацію для кожного БПЛА. Необов’язкове, за замовчуванням true.

У відповідь сервер надсилає запит, приклад якого наведено на рисунку 4.4.



```
{
    "status": "ok",
    "message": "Pre-mission plan was sucsessfully built",
    "vehicle": [
        {
            "name": "b",
            "way": [1, 3, 2, 4, 1],
            "details": [
                " > вилітає з аеропорту a",
                " > знімає ціль 3",
                " > замінює блоки живлення в аеропорту a ",
                " > знімає ціль 2",
                " > замінює блоки живлення в аеропорту a",
                " > знімає ціль 1",
                " > здійснює посадку в аеропорту a"
            ]
        }
    ],
    "score": 272.2,
    "map": "http://uavrp.local/maps/418a-rt5y-po65-5544.html"
}
```


Рисунок 4.4 – приклад відповіді сервера

У разі успішної побудови, код відповіді сервера встановлюється як 200, у випадку помилки – залежно від її типу, згідно з http status codes.

У тілі відповіді є такі поля:

- status – статус виконання завдання, може бути “ok” чи “error”;
- message – короткий опис виконаних дій;
- vehicle – масив, що містить плани обльоту для кожного БПЛА. У полі way вказано ідентифікатори вершин, що мають бути відвідані;
- score – результуюче значення цільової функції;
- map – посилання на отриману карту місцевості.

Слід зазначити, що назва файлу з картою формується у вигляді 16-значного унікального ідентифікатора, а сам файл зберігається щонайменше протягом доби.

У разі помилки сервер поверне відповідь з релевантним http кодом, статусом «error» та коротким описом помилки.

Приклад такої відповіді наведено на рисунку 4.5.

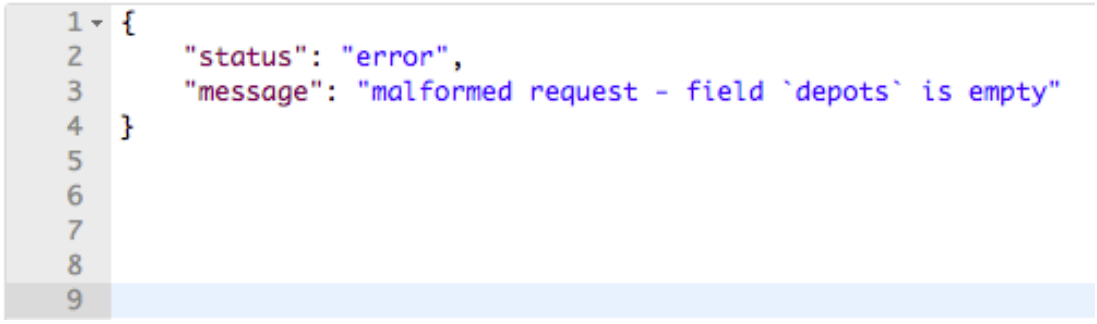


Рисунок 4.5 – приклад відповіді у разі помилки.

5 5 РЕЗУЛЬТАТИ ЧИСЛОВИХ ЕКСПЕРИМЕНТІВ

5.1 Опис проведених досліджень

Кожен з розроблених алгоритмів запущено вказану у кожному експерименті кількість разів на таких задачах:

- проста задача з 1 базою та 3 цілями, 1 БПЛА має запас ходу, достатній для відвідування лише однієї цілі за один виліт. Перевіряє простий план операції на виконання обмежень;
- задача з 3 депо, 15 цілями та 3 БПЛА. Запасу ходу жодного БПЛА не вистачає на повний обліт усіх цілей без заміни блоків живлення;
- задача з 4 депо, 23 цілями та 3 БПЛА. Запасу ходу жодного БПЛА не вистачає на повний обліт усіх цілей без заміни блоків живлення, можна візуально виділити три кластери цілей;
- модифікація задачі att48 з TSPLIB, одна з вершин визначена як депо. 47 цілей, 1 депо та 1 БПЛА;
- модифікація задачі berlin52 з TSPLIB, одна з вершин визначена як депо. 51 ціль, 1 депо та 1 БПЛА.

Для кожної задачі, окрім першої, було виконано попередній підбір параметрів мурашиного алгоритму за допомогою пришвидшених запусків з меншою кількістю ітерацій. З обраними параметрами виконана зазначена кількість запусків з різними ініціалізаторами генератора псевдовипадкових чисел.

Для алгоритму детермінованого локального пошуку у якості початкового розв’язку обирався результат роботи жадібного алгоритму. В усіх випадках ДЛП запускався лише один раз. Час роботи кожного алгоритму в усіх запусках обмежено 20 секундами.

Для обчислення відносної похибки ε використовується f^* – найменше відоме значення цільової функції для задачі, а відносна похибка розв'язку обчислюється за формулою $\varepsilon = 100 \cdot (f - f^*) / f^*$.

5.2 Експеримент 1. Дослідження ефективності розв'язання задачі з 1 базою та 3 цілями

На вхід подаються цілі з наступними координатами:

Широта	Довгота
50.343978	30.850834
50.341835	30.933385
50.363975	30.896167

Координати депо:

Широта	Довгота
50.341315	30.89417

Характеристики БПЛА:

Назва	Запас ходу, км
b	10

Особливістю задачі є те, що запасу ходу вистачає лише на переліт до будь-якої з цілей та назад у депо. Таким чином, треба здійснити дві заміни блоків живлення у депо.

Найкращий відомий план зображено на рисунку 5.1.

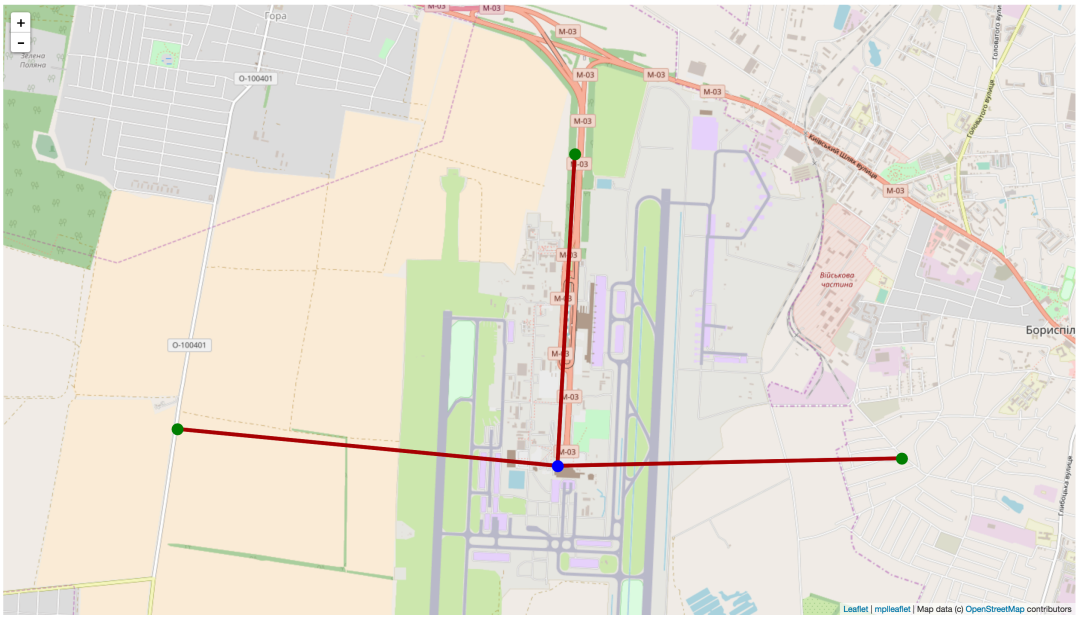


Рисунок 5.1 – отриманий план операції на карті

Оскільки це єдиний можливий маршрут, всі алгоритми повернули однакові результати.

Алгоритм	Класичний ОМК	Новий ОМК	ДЛП
Цільова функція f, км	22,6741	22,6741	22,6741
Відносна похибка ε, %	0	0	0
Час, с	15	15	0.01

На рисунку 5.2 наведено отриманий план операції. Як і очікувалося, відбуваються дві заміни блоків живлення.

Слід також зауважити, що критерієм зупинки алгоритмів на основі ОМК вибрано завершення відведеного на виконання часу, оскільки не встановлено критерій, за яким можна вважати отриманий розв’язок оптимальним.

БПЛ b:

> вилітає з аеропорту a

> знімає ціль 3

> замінює блоки живлення в аеропорту a

> знімає ціль 2

> замінює блоки живлення в аеропорту a

> знімає ціль 1

> здійснює посадку в аеропорту a

Рисунок 5.2 – отриманий план операції у текстовому вигляді

5.3 Експеримент 2. Дослідження ефективності розв'язання задачі з 3 депо, 15 цілями та 3 БПЛА

На вхід подаються цілі з наступними координатами:

Широта	Довгота
50.478404	30.506887
50.498841	30.523816
50.508222	30.630811
50.460862	30.649951
50.379307	30.573624
50.334877	30.571261
50.363888	30.577941
50.415803	30.612011
50.41085	30.576544
50.382149	30.664148
50.374409	30.615389

50.376951	30.949898
50.303762	30.852284
50.393498	30.795309
50.350858	30.95339

Координати депо:

Назва	Широта	Довгота
1	50.36233	30.44366
2	50.447133	30.455596
3	50.341315	30.89417

Характеристики БПЛА:

Назва	Запас ходу, км
a	100
b	80
c	70

Особливістю задачі є те, що точки явно скупчені у два кластери: один біля депо 1 та 2, інший біля депо 3.У найкращому відомому плані операції виліт здійснюється з одного депо, повернення – в інше. Це демонструє описану можливість будувати шляхи між різними депо.

Слід зазначити, що це також актуально для запуску БПЛА з пересувних позицій, тобто, коли повернення не може бути здійснене у ту саму точку.

Найкращий відомий план зображено на рисунку 5.4.

План, отриманий класичним ОМК, зображено на рисунку 5.3.

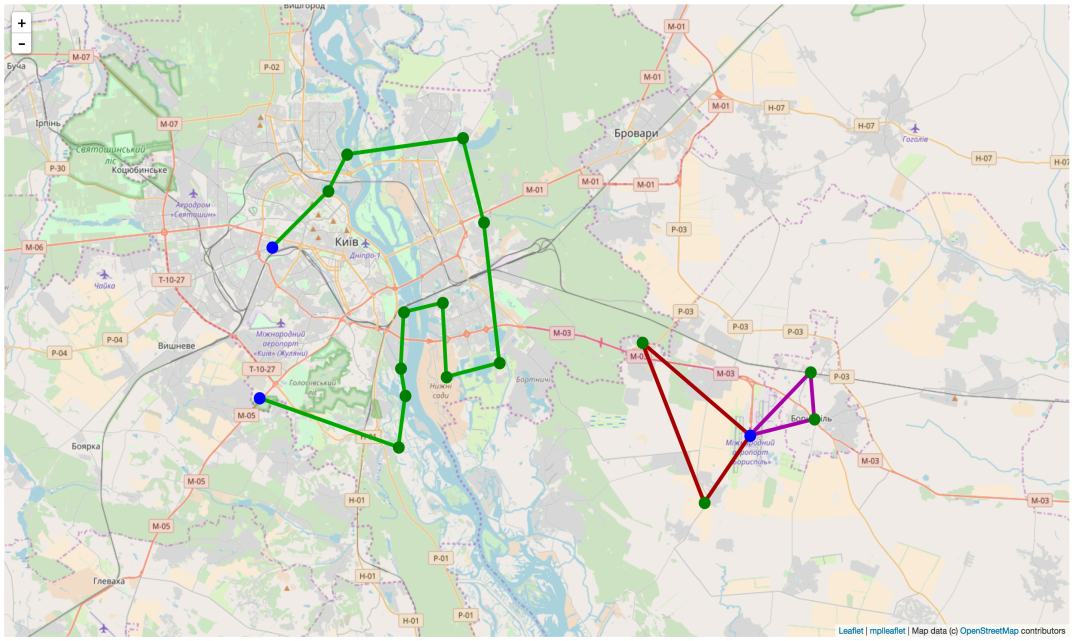


Рисунок 5.3 – отриманий класичним ОМК план операції на карті

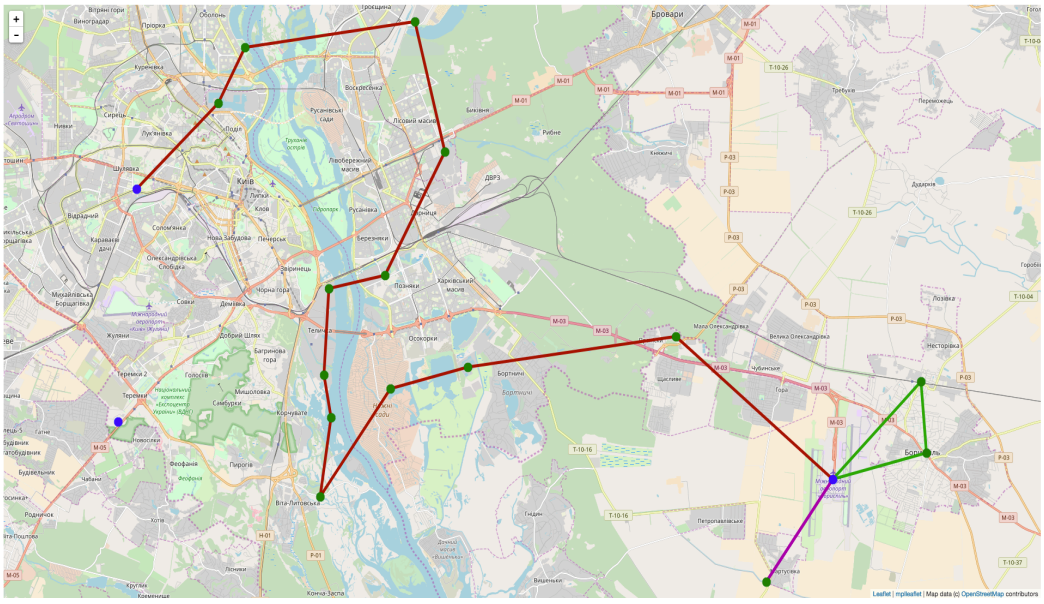


Рисунок 5.4 – отриманий новим ОМК план операції на карті

Класичний та новий ОМК знайшли близькі до найкращого відомого розв’язки. Новий ОМК знайшов найкращий відомий на даний момент розв’язок, в той час як відхилення ДЛП від оптимуму виявилося досить значним.

Алгоритм	Класичний ОМК	Новий ОМК	ДЛП
Цільова функція f, км	113,809	109,738	134.853
Відносна похибка ε, %	3	0	10
Час, с	15	15	0.01

На рисунку 5.5 наведено отриманий ДЛП план операції, що виявився на 10% гіршим за найкращий.

Слід також зауважити, що критерієм зупинки алгоритмів на основі ОМК вибрано завершення відведеного на виконання часу, оскільки не встановлено критерій, за яким можна вважати отриманий розв’язок оптимальним.

Текстовий план, що відповідає найкращому розв’язку, зображено на рисунку 5.6.

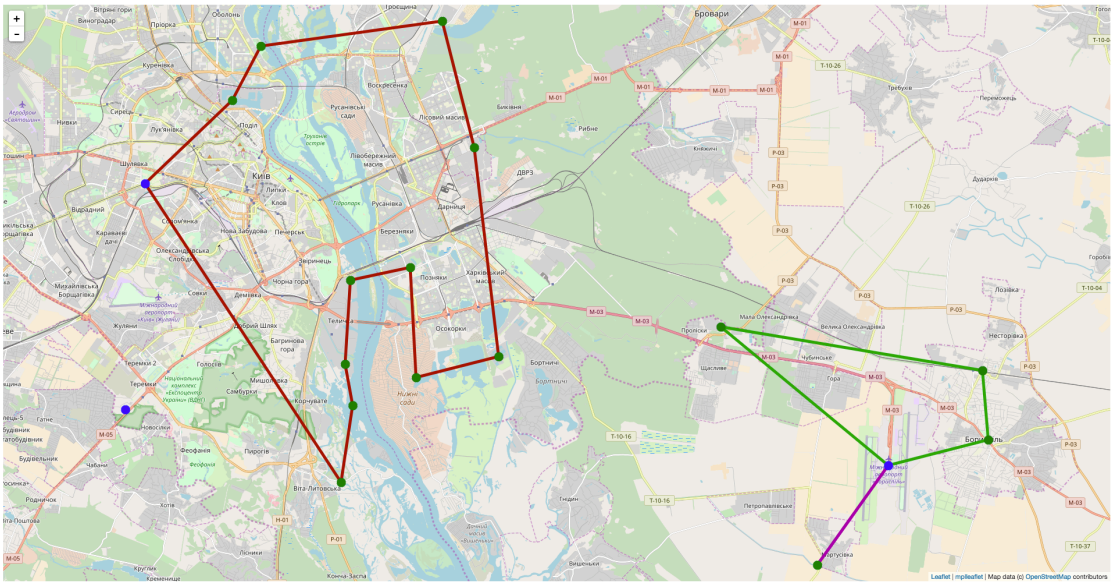


Рисунок 5.5 – план операції на карті, отриманий ДЛП

БПЛ a:

> вилітає з аеропорту 3

> знімає ціль 17

> знімає ціль 16

> здійснює посадку в аеропорту 3

БПЛ b:

> вилітає з аеропорту 3

> знімає ціль 15

> знімає ціль 18

> здійснює посадку в аеропорту 3

БПЛ c:

> вилітає з аеропорту 2

> знімає ціль 4

> знімає ціль 5

> знімає ціль 6

> знімає ціль 7

> знімає ціль 13

> знімає ціль 14

> знімає ціль 11

> знімає ціль 12

> знімає ціль 8

> знімає ціль 10

> знімає ціль 9

> здійснює посадку в аеропорту 1

Рисунок 5.6 – отриманий план операції у текстовому вигляді

5.4 Експеримент 3. Дослідження ефективності розв'язання задачі з 4 депо, 23 цілями та 3 БПЛА

На вхід подаються цілі з наступними координатами:

Широта	Довгота
50.478404	30.506887
50.498841	30.523816
50.508222	30.630811
50.460862	30.649951
50.379307	30.573624
50.334877	30.571261
50.363888	30.577941
50.415803	30.612011
50.41085	30.576544
50.382149	30.664148
50.472196	30.550703
50.478328	30.506661
50.459524	30.525521
50.460423	30.415929
50.43933	30.404943
50.426451	30.373287
50.374409	30.615389
50.376951	30.949898
50.303762	30.852284
50.350858	30.95339
50.536681	30.84932
50.687398	31.111887
50.369928	31.322535

Координати депо:

Назва	Широта	Довгота
A	50.36233	30.44366
B	50.447133	30.455596
C	50.341315	30.89417
D	50.511765	31.040994

Характеристики БПЛА:

Назва	Запас ходу, км
a	70
b	80
c	90

Особливістю задачі є те, що точки явно скупчені у два кластери: один біля депо A та B, інший біля депо C. Також є точки неподалік від депо 4 та 3, проте розкидані на великій відстані одна від одної. У найкращому відомому плані операції є такий маршрут, що виліт здійснюється з одного депо, повернення – в інше. Це демонструє описану можливість будувати шляхи між різними депо.

Також двічі використовується процедура заміни блоків живлення.

Ця задача є показовою з точки зору розкриття особливостей описаної у даній дисертації проблеми.

Найкращий відомий план зображено на рисунку 5.8.

План, отриманий класичним ОМК, зображено на рисунку 5.7.

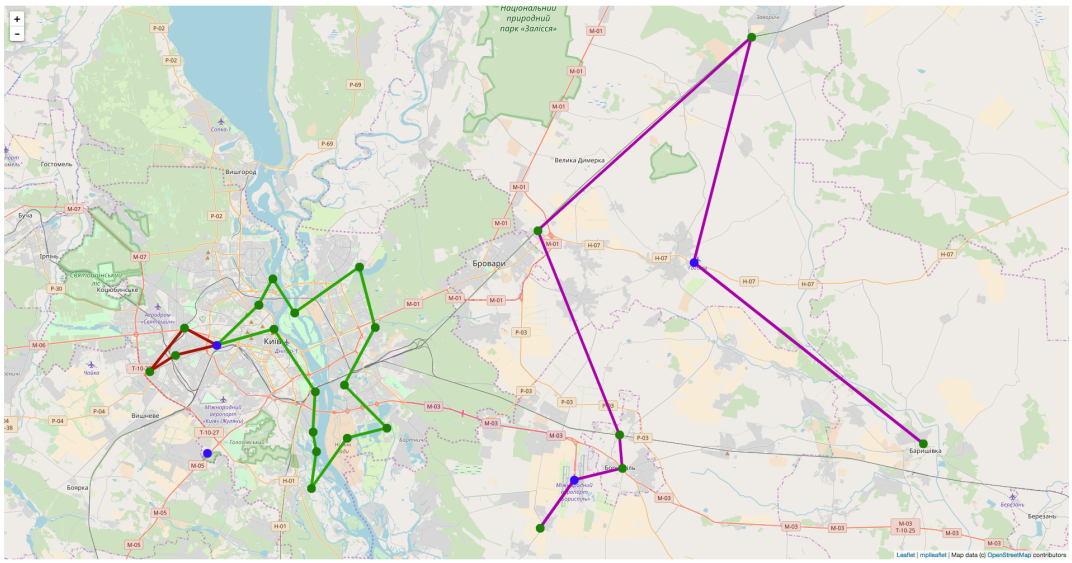


Рисунок 5.7 – отриманий класичним ОМК план операції на карті

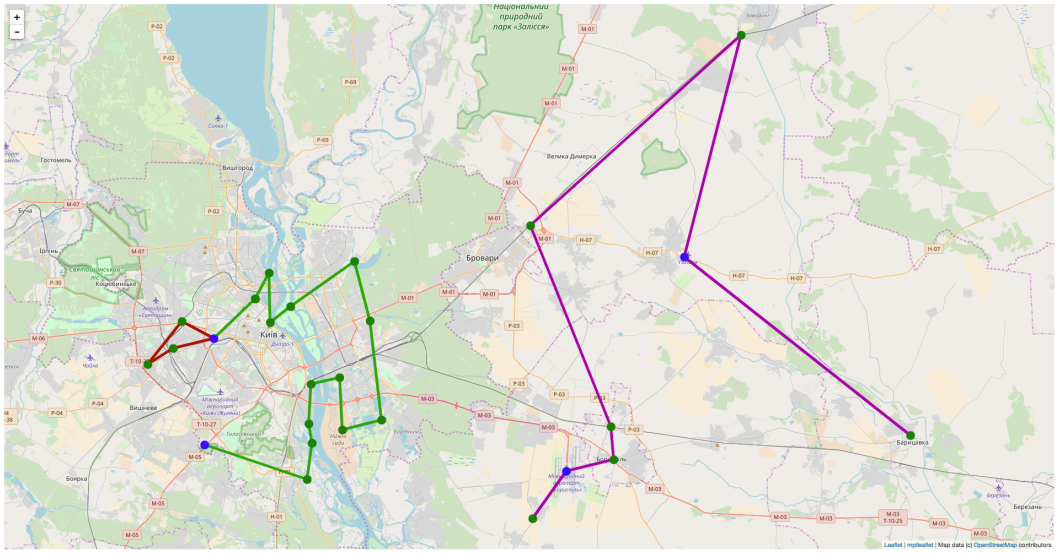


Рисунок 5.8 – отриманий новим ОМК план операції на карті

Класичний та новий ОМК знайшли близькі до найкращого відомого розв’язки. Новий ОМК знайшов найкращий відомий на даний момент розв’язок, в той час як відхилення ДЛП від оптимуму виявилося досить значним.

Алгоритм	Класичний ОМК	Новий ОМК	ДЛП
Цільова функція f, км	253.098	252.021	280.777

Відносна похибка ε, %	0.5	0	11
Час, с	15	15	0.01

На рисунку 5.5 наведено отриманий ДЛП план операції, що виявився на 11% гіршим за найкращий.

Слід також зауважити, що критерієм зупинки алгоритмів на основі ОМК вибрано завершення відведеного на виконання часу, оскільки не встановлено критерій, за яким можна вважати отриманий розв'язок оптимальним.

Текстовий план, що відповідає найкращому розв'язку, зображено на рисунку 5.10.

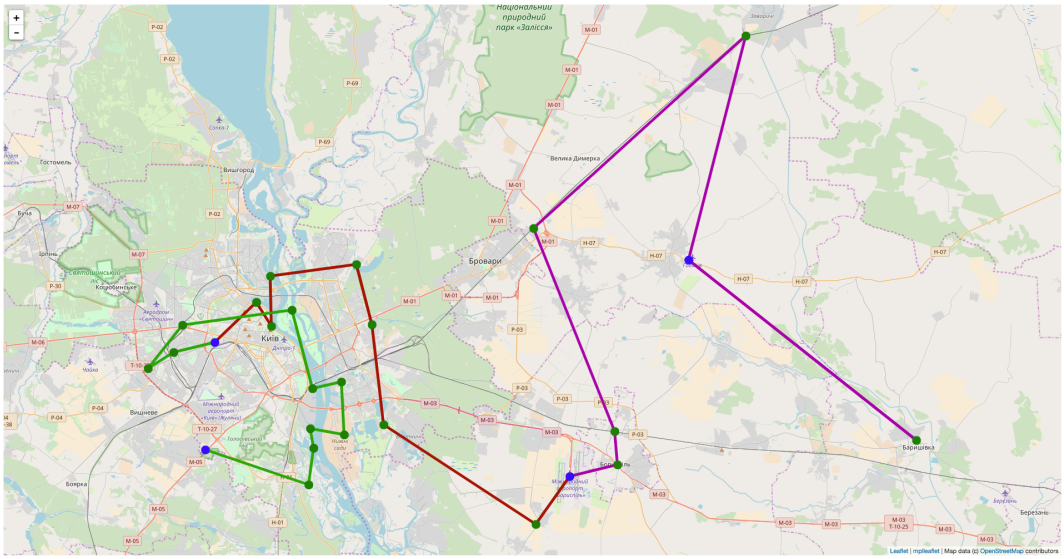


Рисунок 5.9 – план операції на карті, отриманий ДЛП

БПЛ a:

> вилітає з аеропорту B

> знімає ціль 19

> знімає ціль 20

> знімає ціль 18

> здійснює посадку в аеропорту B

БПЛ b:

> вилітає з аеропорту C

> знімає ціль 23

> замінює блоки живлення в аеропорту C

> знімає ціль 24

> знімає ціль 22

> знімає ціль 25

> знімає ціль 26

> замінює блоки живлення в аеропорту D

> знімає ціль 27

> здійснює посадку в аеропорту D

БПЛ c:

> вилітає з аеропорту B

> знімає ціль 17

> знімає ціль 5

> знімає ціль 6

> знімає ціль 16

> знімає ціль 15

> знімає ціль 7

Рисунок 5.10 – отриманий план операції у текстовому вигляді

5.5

5.6 Експеримент 4. Дослідження ефективності розв'язання задачі з 1 депо, 47 цілями та 1 БПЛА

У даній задачі використано вхідні дані для задачі комівояжера, проте одна з цілей була розглянута у якості депо.

Замість геолокацій було подано пари координат X, Y, які було поділено на 100 з метою візуального розміщення на карті.

Метою такого запуску є перевірка алгоритму на задачах з відомої бібліотеки.

Слід зазначити, що використовується евклідова функція відстані замість гаверсинусної.

На вхід подаються цілі з наступними координатами:

X	Y
22.33	0.1
55.3	14.24
4.01	8.41
30.82	16.44
76.08	44.58
75.73	37.16
72.65	12.68
68.98	18.85
11.12	20.49
54.68	26.06
59.89	28.73
47.06	26.74
46.12	20.35
63.47	26.83
61.07	6.69

76.11	51.84
74.62	35.9
77.32	47.23
59	35.61
44.83	33.69
61.01	11.1
51.99	21.82
16.33	28.09
43.07	23.22
6.75	10.06
75.55	48.19
75.41	39.81
31.77	7.56
73.52	45.06
75.45	28.01
32.45	33.05
64.26	31.73
46.08	11.98
0.23	22.16
72.48	37.79
77.62	45.95
73.92	22.44
34.84	28.29
62.71	21.35
49.85	1.4
19.16	15.69
72.8	48.99
75.09	32.39

0.1	26.76
68.07	29.93
51.85	32.58
30.23	19.42

Координати депо:

Назва	X	Y
A	67.34	14.53

Характеристики БПЛА:

Назва	Запас ходу, км
a	9000000

Особливістю задачі є те, що задача комівояжера з відомим оптимумом перетворюється на задачу маршрутизації БПЛА шляхом заміни однієї вершини на депо. Оптимальний шлях залишається незмінним, а отже, є можливість порівнювати отриманий розв’язок з найкращим.

Ця задача також відома як att48 з TSPLIB.

Найкращий відомий план зображено на рисунку 5.11.

Рисунок 5.11 – оптимальний розв'язок задачі att48 на карті

Класичний та новий ОМК знайшли оптимальні розв'язки, в той час як відхилення ДЛП від оптимуму виявилося досить значним.

Алгоритм	Класичний ОМК	Новий ОМК	ДЛП
Цільова функція f, км	10628	10628	11523
Відносна похибка ε, %	0	0	8.4
Час, с	15	15	0.01

Слід також зауважити, що критерієм зупинки алгоритмів на основі ОМК вибрано завершення відведеного на виконання часу, оскільки не встановлено критерій, за яким можна вважати отриманий розв'язок оптимальним.

Текстовий план, що відповідає найкращому розв'язку, зображено на рисунку 5.12.

БПЛ plane:

> вилітає з аеропорту 1
> знімає ціль 8
> знімає ціль 38
> знімає ціль 31
> знімає ціль 44
> знімає ціль 18
> знімає ціль 7
> знімає ціль 28
> знімає ціль 6
> знімає ціль 37
> знімає ціль 19
> знімає ціль 27
> знімає ціль 17
> знімає ціль 43
> знімає ціль 30
> знімає ціль 36
> знімає ціль 46
> знімає ціль 33
> знімає ціль 20
> знімає ціль 47
> знімає ціль 21
> знімає ціль 32
> знімає ціль 39
> знімає ціль 48
> знімає ціль 5
> знімає ціль 42
> знімає ціль 24
> знімає ціль 10
> знімає ціль 45
> знімає ціль 35

Рисунок 5.12 – отриманий план операції у текстовому вигляді (початок)

5.7

> знімає ціль 4
> знімає ціль 26
> знімає ціль 2
> знімає ціль 29
> знімає ціль 34
> знімає ціль 41
> знімає ціль 16
> знімає ціль 22
> знімає ціль 3
> знімає ціль 23
> знімає ціль 14
> знімає ціль 25
> знімає ціль 13
> знімає ціль 11
> знімає ціль 12
> знімає ціль 15
> знімає ціль 40
> знімає ціль 9
> здійснює посадку в аеропорту 1

Рисунок 5.13 – отриманий план операції у текстовому вигляді (продовження)

5.8 Експеримент 5. Дослідження ефективності розв'язання задачі з 1 депо, 51 ціллю та 1 БПЛА

У даній задачі використано вхідні дані для задачі комівояжера, проте одна з цілей була розглянута у якості депо.

Замість геолокацій було подано пари координат X, Y, які було поділено на 100 з метою візуального розміщення на карті.

Метою такого запуску є перевірка алгоритму на задачах з відомої бібліотеки.

Слід зазначити, що використовується евклідова функція відстані замість гаверсинусної.

На вхід подаються цілі з наступними координатами:

X	Y
0.25	1.85
3.45	7.5
9.45	6.85
8.45	6.55
8.8	6.6
0.25	2.3
5.25	10
5.8	11.75
6.5	11.3
16.05	6.2
12.2	5.8
14.65	2
15.3	0.05
8.45	6.8
7.25	3.7
1.45	6.65
4.15	6.35
5.1	8.75
5.6	3.65
3	4.65
5.2	5.85
4.8	4.15
8.35	6.25
9.75	5.8

12.15	2.45
13.2	3.15
12.5	4
6.6	1.8
4.1	2.5
4.2	5.55
5.75	6.65
11.5	11.6
7	5.8
6.85	5.95
6.85	6.1
7.7	6.1
7.95	6.45
7.2	6.35
7.6	6.5
4.75	9.6
0.95	2.6
8.75	9.2
7	5
5.55	8.15
8.3	4.85
11.7	0.65
8.3	6.1
6.05	6.25
5.95	3.6
13.4	7.25
17.4	2.45

Координати депо:

Назва	X	Y
A	5.65	5.75

Характеристики БПЛА:

Назва	Запас ходу, км
a	9000000

Особливістю задачі є те, що задача комівояжера з відомим оптимумом перетворюється на задачу маршрутизації БПЛА шляхом заміни однієї вершини на депо. Оптимальний шлях залишається незмінним, а отже, є можливість порівнювати отриманий розв’язок з найкращим.

Ця задача також відома як berlin52 з TSPLIB.

Найкращий відомий план зображено на рисунку 5.14.

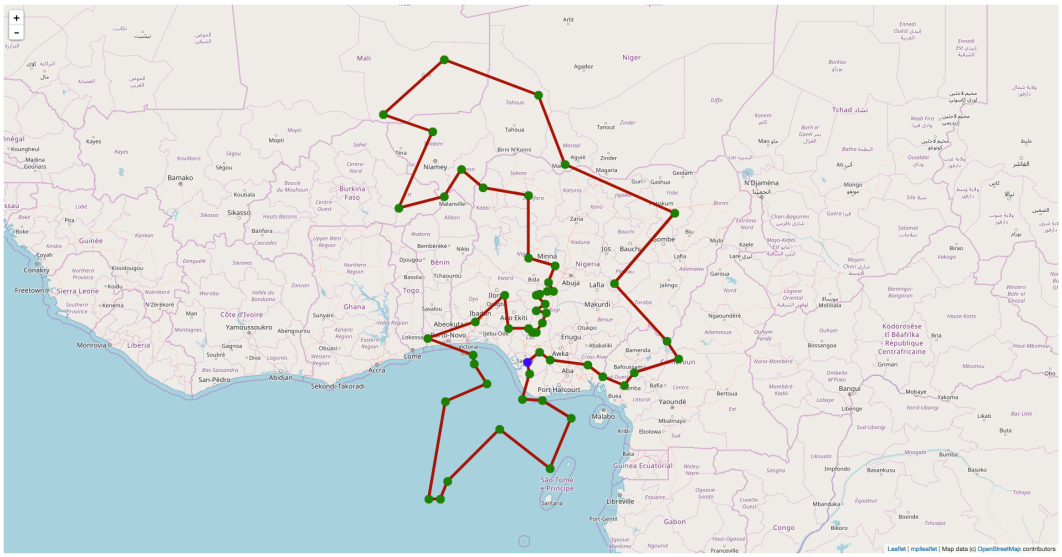


Рисунок 5.14 – оптимальний розв’язок задачі att48 на карті

Класичний та новий ОМК знайшли оптимальні розв’язки, в той час як відхилення ДЛП від оптимуму виявилося досить значним.

Алгоритм	Класичний ОМК	Новий ОМК	ДЛП
Цільова функція f, км	7542	7542	8127
Відносна похибка ε, %	0	0	7.7
Час, с	15	15	0.01

Слід також зауважити, що критерієм зупинки алгоритмів на основі ОМК вибрано завершення відведеного на виконання часу, оскільки не встановлено критерій, за яким можна вважати отриманий розв’язок оптимальним.

Текстовий план, що відповідає найкращому розв’язку, зображено на рисунку 5.16.

Результат роботи ДЛП показано на рисунку 5.15.

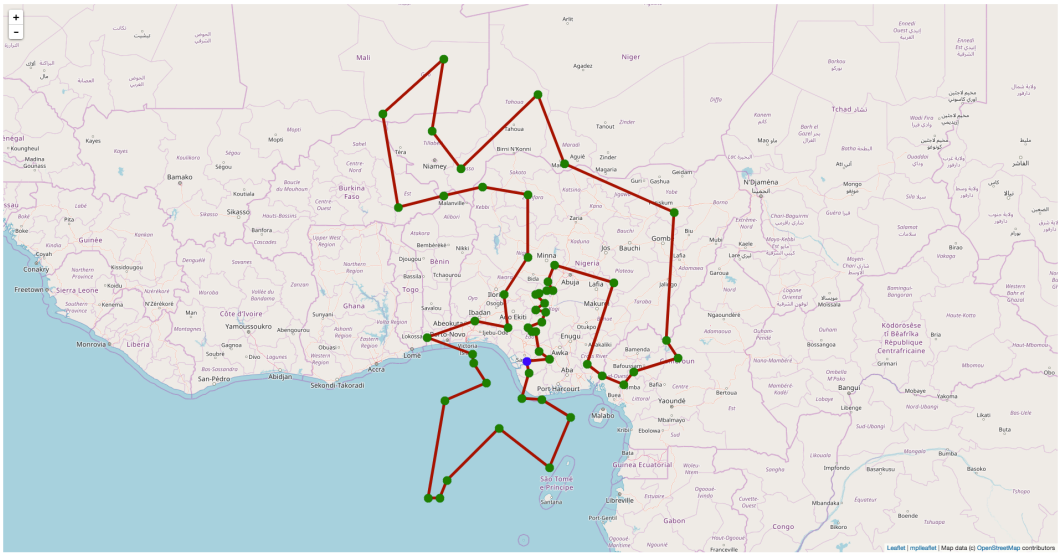


Рисунок 5.15 – результат, отриманий за допомогою ДЛП

БПЛ plane:

> вилітає з аеропорту 1
> знімає ціль 49
> знімає ціль 32
> знімає ціль 45
> знімає ціль 19
> знімає ціль 41
> знімає ціль 8
> знімає ціль 9
> знімає ціль 10
> знімає ціль 43
> знімає ціль 33
> знімає ціль 51
> знімає ціль 11
> знімає ціль 52
> знімає ціль 14
> знімає ціль 13
> знімає ціль 47
> знімає ціль 26
> знімає ціль 27
> знімає ціль 28
> знімає ціль 12
> знімає ціль 25
> знімає ціль 4
> знімає ціль 6
> знімає ціль 15
> знімає ціль 5

5.9 Рисунок 5.16 – отриманий план операції у текстовому вигляді (початок)

5.10

> знімає ціль 24
> знімає ціль 48
> знімає ціль 38
> знімає ціль 37
> знімає ціль 40
> знімає ціль 39
> знімає ціль 36
> знімає ціль 35
> знімає ціль 34
> знімає ціль 44
> знімає ціль 46
> знімає ціль 16
> знімає ціль 29
> знімає ціль 50
> знімає ціль 20
> знімає ціль 23
> знімає ціль 30
> знімає ціль 2
> знімає ціль 7
> знімає ціль 42
> знімає ціль 21
> знімає ціль 17
> знімає ціль 3
> знімає ціль 18
> знімає ціль 31
> знімає ціль 22
> здійснює посадку в аеропорту 1

Рисунок 5.17 – отриманий план операції у текстовому вигляді (продовження)

5.11

5.12 Аналіз результатів числових експериментів

Розроблені алгоритми було апробовано як на відомих задачах комівояжера, так і на спеціально розроблених задачах з багатьма депо і строгими обмеженнями. Новий алгоритм у більшості випадків демонструє кращий результат за розроблений звичайний ОМК, проте, останнє покращення здійснюється ним значно пізніше.

Різниця між новим та звичайним алгоритмом на задачах комівояжера не проявлялася, проте, новий алгоритм забезпечує значно більшу варіативність розглянутих розв’язків, ніж звичайний. Особливо це впливає на вибір початкової розстановки БПЛА по депо.

У якості підсумкової рекомендації можна зазначити, що новий алгоритм добре підходить у випадку, коли немає жорстких обмежень на час побудови плану операції. На задачах невеликої розмірності (до 52, згідно тестових запусків), різниця у часі є нехтовно малою.

ЛІТЕРАТУРА

1. Kamil A. Alotaibi (2014). Unmanned Aerial Vehicle Routing In The Presence Of Threats. Arlington: The University Of Texas At Arlington. 25-121.

1a. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. – К.: Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с.

1b. Маршрутизация полета легкого беспилотного летательного аппарата в поле постоянного ветра на основе решения разновидностей задачи коммивояжера (Моисеев Д. В. +)

1c.https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/8860/1/%D1%82%D0%B5%D0%B7%D0%B8.pdf

1d. Pintea C.M. Advances in Bio-inspired Computing for Combinatorial Optimization Problems / C.M. Pintea - Berlin, Heidelberg: Springer-Verlag, 2014. - 188 p.

1e. Гуляницький Л.Ф. Новий алгоритм оптимізації мурашиними колоніями. Пр. Міжн. конф., присвяченої 60-річчю заснування ІК ім.В.М.Глушкова НАН України "Сучасна інформатика: проблеми, досягнення та перспективи розвитку" (Київ, 13-15 грудня 2017). К.: ІК ім.В.М.Глушкова НАН України, 2017. С. 41-43.

2. Garey, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. New York, USA: W.H. Freeman & Co Ltd.

3. Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59, 345–358.

4. Baldacci, R., Toth, P., & Vigo, D. (2010). Exact algorithms for routing problems under vehicle capacity constraints. Annals of Operations Research, 175, 213–245

5. Solomon, M. M., & Desrosiers, J. (1988). Time window constrained routing and scheduling problems. Transportation Science, 22(1), 1–13.

6. Renaud, J., Laporte, G., & Boctor, F. F. (1996). A Tabu search heuristic for the multidepot vehicle routing problem. Computers & Operations Research, 23(3), 229–235

7. Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. In B. L. Golden, S. Raghavan, & E. Wasil (Eds.), The vehicle routing problem: latest advances and new challenges (pp. 3–27). Berlin: Springer. Vol. 43.

8. https://prozorro.gov.ua/tender/UA-2017-11-06-001777-c

9. https://prozorro.gov.ua/tender/UA-2017-07-11-000444-c

10. “A literature review on the vehicle routing problem with multiple depots”

11. Glen van Brummelen. Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry By Glen Van Brummelen

12. Пожидаев М. С. Алгоритмы решения задачи маршрутизации транспорта [Текст] : дис. … канд. техн. наук: 05.13.18 / М. С. Пожидаев. – Томск: ГОУВПО «Томский государственный университет», 2010. – 136 с.

13. Штовба С. Д. Муравьиные алгоритмы: теория и применение [Текст] / С. Д. Штовба // Программирование. – 2005. – № 4. – С. 1–16.

14. Штовба С. Д. Мурашині алгоритми оптимізації [Текст] / С. Д. Штовба, О. М. Рудий // Вісник ВПІ. – 2004. – № 4. – С. 62–69.

15. Chiang W. Simulated annealing metaheuristics for the vehicle routing problem with time windows / Chiang W. & Russell R.A. // Annals of Operations Research. – 1996. – № 63. – pp. 3–27.

16. Braysy O. Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows / Braysy O., Dullaert W. & Gendreau M. // Journal of Heuristics. – 2004. – № 10. – pp. 587–611.

17. Nagy G. Location-routing: Issues, models and methods / Nagy G. & Salhi S. // European Journal of Operational Research. – 2007. – № 177. – pp. 649–672.

18. Choi E. A column generation approach to the heterogeneous fleet vehicle routing problem / Eunjeong Choia, Dong-Wan Tcha // Computers & Operations Research. – 2007. [Електронний ресурс] Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2854&rep=rep1 &ty pe=pdf

19. Laporte G. Routing problems: A bibliography / Laporte G. & Osman I.H. // Annals of Operations Research. – 1995. – № 61. – pp. 227–262.

20. Taillard R.E. Parallel iterative search methods for vehicle routing problems // Networks. – 1993. – № 23. – pp. 661–73. 21. Rochat Y. Probabilistic diversification and intensification in local search for vehicle routing / Rochat Y. & Taillard R.E. // Journal of Heuristics. – 1995. – № 1. – pp. 147–167. 22. Renaud J. An improved petal heuristic for the vehicle routing problem / Renaud J., Boctor F. F. & Laporte G. // Journal of the Operational Research Society. - 1996. - № 47. – pp. 329–336.

21. Rochat and Taillard 1995: Probabilistic diversification and intensification in local search for vehicle routing

22. Van Brummelen G. Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry. Princeton, N.J., Oxford: Princeton University Press, 2013. P. 160-162.

23. https://internetdevels.ua/blog/building-microservices-dotnet

24. http://it-ua.info/news/2016/02/17/pdhodi-do-proektuvannya-restful-api.html